(2)

DEVELOPMENT OF SOFTWARE FOR THE BASE-LEVEL WAR RESERVE MATERIELS WRM PROGRAM

THESIS

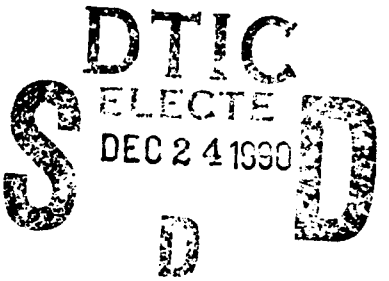
Kevin M. Tinker, Captain, USAF

AFIT/GLM/LSM/90S-58

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GLM/LSM/90S-58



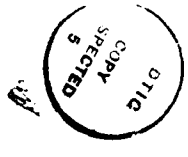
DEVELOPMENT OF SOFTWARE FOR THE BASE-LEVEL WAR RESERVE MATERIELS (WRM) PROGRAM

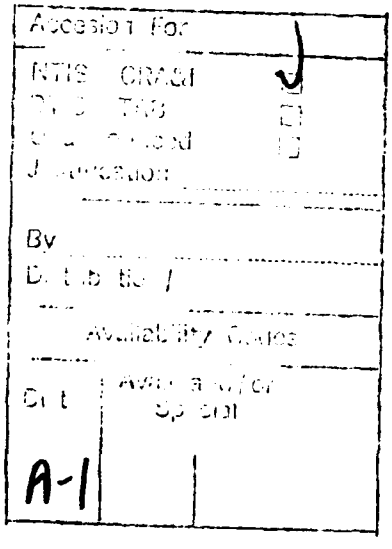
THESIS

Kevin M. Tanzer, Captain, USAF

AFIT/GLM/LSM/90S-58

The opinions and conclusions in this paper are those of the author and are not intended to represent the official position of the DOD, USAF, or any other government agency.





AFIT/GLM/LSM/90S-58

DEVELOPMENT OF SOFTWARE FOR THE BASE-LEVEL WAR RESERVE MATERIELS (WRM) PROGRAM

THESIS

Presented to the Faculty of the School of
Systems and Logistics
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Masters of Science in Logistics Management

Kevin M. Tanzer, B.A.
Captain, USAF

September 1990

Preface

The purpose of this research was to decide if a personal computer application was useful for managing War Reserve Materiel (WRM) information. The research required interviews with several expert logisticians in offices throughout the Air Force. Based on the requirements of the experts, a database management system application was developed. The computer application was then evaluated by a group of users. The program was well received, and was accepted by 86.7 % of the users as a useful tool. The program should be further developed, expanded, and evaluated in an operational environment.

In performing the research and writing of this thesis I had the support of several people. I am indebted to my advisor, Major Phil Beard for his patience and assistance, particuarly the software utilities used to capture images of the computer program screens. I also wish to thank my friend Susan M. Coller for her understanding and patience on the many evenings and weekends I had to work at my desk.

Kevin M. Tanzer

Table of Contents

List of Figures

Abstract

This study investigated the requirements for development of a software program for managing War Reserve Materiel (WRM) consumable assets. Areas examined included the requirements for collection, organization, and presentation of data. The research also examined which computer language was appropriate for the program, interface requirements, criteria for successful implementation, and acceptance by the users. A literature review revealed numerous instances of software programs researched and developed to permit management of other WRM program information. The research followed a ten step methodology for developing a database management system application. As part of the methodology, the researcher interviewed eleven experts to determine program requirements. The study developed the prototype software, which was then evaluated by an expert. Following correction of program flaws, a final version of the prototype software was develop. The program was then delivered to fifteen experts for evaluation. The program was accepted by 86.7% of the experts as being a useful tool for managing a WRM program.

DEVELOPMENT OF SOFTWARE FOR THE BASE-LEVEL WAR RESERVE MATERIELS (WRM) PROGRAM

I. Introduction

General Issue

After the United States withdrew from Vietnam, the individual services recognized the need for readily available equipment and supplies for deploying forces. One way to provide the equipment and supplies is to stockpile them at convenient locations for use by deploying forces. Since equipment and materiels cost money, stockpiling results in trading cost for time -- the time it would take to overcome production and delivery lead times.

In the early 1970's, the Army began an aggressive program in Europe, called POMCUS (Pre-positioning of Materiel Configured Unit Sets) with the aim of having fully combat capable equipment available to arriving support forces from the United States. The Navy began a similar program in 1979, stationing 12 ships at Diego Garcia in the Indian Ocean with enough supplies and equipment to support three marine amphibious brigades (23:281). The Air Force also recognized a need for pre-positioning of materiels overseas, and set up the War Readiness Materiel program in the mid 1960's. Many problems existed with the system: incorrect inventory balances, wrong items included in the

program, and serious omissions (4:11). For example, gun barrels and parts were included in the early WRM kits, but not gun oil. Although many of the program problems have been solved, the program is still scrutinized closely at the base level. The most common concerns include actual (on-hand) inventory amounts versus planned (authorized) amounts, availability of some items from the supply system, and the physical condition of items on-hand (20).

The aim of the War Reserve Materiel (WRM) program is to give wartime support for US forces and allies. To meet this objective, the WRM program covers

> ... authorization, acquisition, prepositioning, prestockage, storage and maintenance of all WRM needed to support HQ USAF approved wartime plans and programs (11:1-1).

This charter covers a broad range of assets. There are many classes of WRM assets: support equipment, munitions, WCDO consumable items, vehicles, Harvest Bare, Harvest Eagle, and Harvest Falcon equipment packages. Support equipment includes aircraft repair parts and spares for aircraft. Munitions are missiles, ordnance, explosives and related materials; it also includes chaff and flares. WCDO consumable items are expendable items directly related and necessary for a weapons system or combat support (11:Attch 2). Not all vehicles are WRM items, only those specifically authorized in the Table of Allowance (TA) 010. The Harvest packages support personnel, maintenance and supply in a bare base environment, and are used for deployment of either flying or non-flying units. The Harvest Bare packages

include portable housing, electrical generators, tents, mess hall cooking equipment, and other items needed to set up and maintain an 'instant' base at a prepared sight. Harvest Eagle and Harvest Falcon are similar to Harvest Bare packages, but are tailored in size and composition to support just the personnel needed for deployment of an F-15 or F-16 aircraft package (11:5).

To help manage these items, several different listings, or reports, are available from HQ USAF, the Major Commands (MAJCOM) and from the supply computer system at the bases. The task of reviewing, analyzing, and understanding the listings used at the base level is challenging and needs simplifying (28). Listings generated by computer systems at the MAJCOM headquarters and at the base come in many different formats. The base logistician frequently must go through different reports to extract data elements, and then relate the separate data pieces. There is a need to organize the mass of data presented to the base logistics planner and to present that data in an informative manner (20).

The many assets involved in the War Reserve Materiels program have put a heavy workload on the Logistics Plans and Program office (LGX) at most bases. Logistics planners in the Tactical Air Command (TAC), Pacific Air Forces (PACAF), Strategic Air Command (SAC), and the US Air Forces in Europe (USAFE) spend a considerable amount of time tracking, inspecting, and managing WRM program data (7;15).

The use of a computer is applicable to organizing WRM data because computers have the ability to store large amounts of data, can quickly retrieve data, and are flexible in presenting desired information.

Problem Statement

This study proposes to develop a prototype software program to permit management of WRM asset information, including Pre-Positioned Packages (PPP), War Consumable Distribution Objective (WCDO) items and consumable items in the Wartime Plans Additive Requirements Report (WPARR).

Investigative Questions

To develop an effective software program to track and manage WRM asset information, several investigative questions must be answered:

1. Can a program be developed that will meet requirements identified by logisticians and by Air Force Regulations?
2. What are the functional requirements of a WRM software system for effective collection, organization and presentation of information?
3. What is the appropriate computer language for system development that will be efficient, effective, and still maintain user-friendliness?
4. What form should a database take to efficiently organize WRM data?
5. Will the program need to access and work with existing databases or computer systems?
6. What are the criteria to ensure successful implementation and acceptance by base-level logisticians?

Justification

The Logistics Plans and Program (LGX) office manpower availability is decreasing while the workload increases, therefore the time spent handling information is at a premium. A reduction of the hours spent leafing through reams of paper to piece together data will save the time of the logistics planner. By automating the printing of a specialized report or the search for specific data, it would free the time spent by the base logistician to pursue other tasks. The logistician could use this time to focus on problem areas in the WRM program. With increasing interest in the levels and conditions of spare parts and equipment, the Air Force needs to improve the 'stubby pencil' approach to maintaining information (10:64). The benefit of developing WRM software is a savings of the man-hours spent in managing the WRM program. A user who spends all day managing WRM might save as much as an hour per day. Also, an increase in the accuracy of WRM program data may result in additional savings (5).

Method of Treatment and Organization

Chapter I gives background information, the specific problem, associated investigative questions, limitations, and assumptions of the study. Chapter II reviews the literature in support of the study, covering the characteristics of a database and database management systems, and similar applications of a database management

system. Chapter III outlines the methodology, a requirements determination through a series of telephone interviews, initial program development, review of the program by a user, further program development and validation of the program by many users. Chapter IV contains the data and results of the methodology outlined in Chapter III. It includes recommendations of the on-site visit with the Air Force Logistics Management Center (AFLMC). Chapter V contains the analysis and conclusions from the program review and validation, along with recommendations for further study.

Limitations of the Study

This study does not examine the requirements for handling of classified information, which would require expensive hardware investments by the user and approval of the software developed from this research. In the Air Force, classified data is restricted to use on TEMPEST approved computer systems. Large computer systems must meet similar restrictions. The TEMPEST personal computers are significantly more expensive than non-TEMPEST machines and most LGX offices do not have these machines. Additionally, to gain approval to use the resulting software on a TEMPEST computer would require more time than is available for this study. The resulting software developed from this study could handle classified information on an appropriate system.

Additionally, the software program does not handle every possible type of WRM item. Specialized software programs are available for use with Harvest Eagle, Harvest Bare, and Harvest Falcon packages. The author did not try to duplicate existing software capabilities, such as mainframe software programs (14). An interface with other programs would have complicated program execution, data structures, and imposed security classifications on the software. Also, the software program does not handle every possible manipulation of WRM data, but rather the most strongly desired capabilities expressed during the requirements determination process.

Assumptions

Program development is a continuous process. A program developed to meet user specifications needs more development when the user actually sees and uses the final software. This study assumes that a review of the initial program will identify any major problems or shortfalls with the software and documentation. The final version of the program should then correct those findings.

This study also assumes that the individuals interviewed to develop program requirements are representative of the population of logistics planners; their views represent the views of base logistics planners.

The interview technique used in determining requirements does not allow the users to evaluate the software as a single group. Every user was sent the

software and User's Guide by mail for evaluation. It was assumed that if seventy five percent or more of the respondents accepted the software as helpful in managing a WRM program, the research project was a success. The results of the evaluation were not statistically evaluated because of the small sample of participants.

Summary of Chapter I

The WRM program of the Air Force involves a variety of assets, such as consumables, vehicles, and equipment. Many of the assets are included in different reports, each with different formats. Managing the information in the WRM program is time-consuming, and places a heavy workload on Logistics Plans and Program offices. A computer application is applicable to organizing the data because of the computers ability to retrieve large amounts of data, and organize the data. A series of investigative questions were presented. The study is limited to non-classified data, and did not try to duplicate existing software program capabilities. The study assumed the software developed was a prototype, and that the requirements stated by the logistic planners were representative of the population of all the planners. The study also assumed that a seventy five percent acceptance of the program would represent a successful program development.

II. Literature Review

Introduction

Computers have quickly become an integral tool for the military and are used in many aspects of daily operations. With the increasing availability of computer systems, there has been an increasing demand for computer programs that effectively use computer capabilities. The need for a database management system (DBMS) was identified to help manage War Reserve Materiel assets in the Air Force by the AFLMC at Gunter AFB (28:1).

To ensure that a DBMS application will meet the requirements of the user, the program must have good programming techniques and program structure. This literature review identifies the characteristics of a good database and some goals of a database management system, discusses selection of software, and reviews prior instances where database management system applications have been applied to similar problems.

Discussion of Literature

Database Management System. Databases are used to store information about an item or a subject in an orderly way. Each set of information about the item or subject is called a record. A record can be thought of as a single line of information about an item, such as height, width,

weight, and cost. By adding more records, a database is created. Every time another item is added to the database, another line of information is added with height, width, weight and cost data. A record is also called a node and several nodes, or records, make up a database (18:67). A formal definition of a database is "a collection of interrelated data stored together without harmful or unnecessary redundancy to serve one or more applications in an optimal fashion" (21:19). Martin breaks a database management system into its parts.

> A collection of information gathered together by common criteria is a database. A data item is the smallest unit of data. Data which is grouped together in a named collection is a record. A file is the named collection of records. A database is a collection of records that contains the relationship between records and data items. (21:19)

Databases can be electronic, as in a computer application, or physical; a library card catalog is a database. All cards in the catalog contain the same types of information: name of author, title of the work, publisher, publishing date, and a catalog number. The collection is a database because all the cards are part of the Library of Congress Catalog Number System. This system establishes the relationships among the cards. A practical definition of a database is "a collection of related data about an enterprise with multiple uses" (1:11). A database management system application is then a collection of databases, however, the contents of the different databases

are independent, and not always related to each other. A database management system (DBMS) stores, organizes, and retrieves the data from the databases in a way that makes sense to the user, turning data into information (18:2). Computer databases take one of three different forms: hierarchical, network, or relational (Martin:95).

Hierarchical structure. A hierarchical database structure is similar to the root structure of a tree. The structure consists of 'nodes' or records, connected by 'branches' that link the nodes. The links allow the database to move from one record to the another. The first node is the root node, and will have several nodes connected below it (1:67). The relationship of a node to another is a 'parent' or 'child' where the child node is dependent on the parent node to describe its location in the database. A hierarchical structure allows the database management system to move to any of the records connected below the parent record. If the parent node is removed from the database without connecting the child node to another parent node, the child node is lost, as are all lower nodes. In the hierarchical structure, there is no limit to the number of child nodes or records that a parent node can have. The distinguishing feature of a hierarchical structure is that every node (except the root node) has exactly one parent node (1:69).

Network structure. A network database structure is a complex arrangement of relationships between nodes, where a child node is tied to several parent nodes (1:93). Network database structures can have a string of nodes where the last node is a parent to the first node in the string. This complexity in relationships makes it difficult to change the structure of the database (18:82). The advantage to this method is that it allows the programmer to reflect the complex relationships between the information in the nodes (18:85).

Relational structure. A relational database structure is the most common structure used in database management systems. The form of the structure is a stack, where each node represents a record (21:95). Each node is the parent to a single child node directly below it. Since each node has the same types of data, Martin describes this structure as a table of columns and rows (21:97). The row is a node, while the column represents common data types. A simple example of a database is a table of measurements for several like pieces of equipment. Each piece of equipment has its own row, while all the similar data, like weight, is in the same column. The technical literature refers to a column as an attribute, and a row as a tuple. The most common terms to describe a relational database structure are table, column, and row (21:96).

Characteristics of a Good Database

Data independence. The structure of a database should not be dependent on the type of data stored. This is done by using variable names that represent the data element or column. After removing a data element or column from the structure, the system can still access any node in the structure without referring to the missing data element. The net effect is the ability to use the database without knowing the representative details of the data (1:15).

Data redundancy. When storing the same data repeatedly in the records or nodes of a database, it is redundant data. If the data results in faster recall, it is beneficial to database operation. As the number of records increases, however, the repeated storage of the data takes additional memory, and can slow down the processing time significantly (12:20). The storage of the same data many times over also increases the size of the database, and uses more of the storage media. By storing the repeated information in a separate table or database and recalling it only as needed by the program, data redundancy is kept to a minimum (1:45).

Data Integrity. Data integrity is the process of "ensuring that the updates (of data) are correct, even during failure periods" (21:63). To preserve data integrity, the programmer must plan for making backups of the data and protecting against input errors from the user. The programmer must also plan for catastrophic damage to the

program and data files and restoration of the damaged files (21:64). The most common backup methods are making copies of all necessary files on a scheduled basis, and continuous backups of information from changes. To protect against input errors, the program should check the type of data entered against the type of data it expects (12:56). For example, if the program is asking for a social security number, the program should not allow alphabetic characters. The program should also check the value of data entered to ensure it is does not exceed expected ranges. For example, the number of supply items ordered should not be a negative number, nor should it be a zero quantity. Similar checking should be done for calendar dates. If the program is asking for the name of a person to search for, it should not allow the user to enter a number.

Protection against catastrophic damage, typically fire or water damage, is dependent on the facility having the database management system (12:54). The programmer can plan for recovery of the system if such a failure occurs through a separate program that re-establishes the software when run, and prompts the user for the most recent backup files. Elbra suggests that one means of restoring a database is reading in a backup copy of the database. Alternatively, a running log of changes to the database since the last backup can update the database again, and thereby restore the system to its exact state before the system failure (12:55).

Objectives of a Database

The development of a database shares several common objectives with the development of a database management system as described by Atre. The programmer should consider using the same database for several program functions, eliminating data redundancy and preserving data independence, and data integrity (1:13). Another element to consider is the ease of use in adding, editing, deleting, and retrieving data. Most database management systems allow the user to perform the four basic functions of adding data, editing existing data, deletion of data, and creation of reports or listings of data (16:189).

Selection of software

The author has experience with several database management systems and programming languages. The database software included Enable and Condor III, as well as dBASE II, dBASE III and dBASE III Plus (tm). The programming languages have included BASIC, FORTRAN, C, and Pascal. Of the database software and languages, dBASE III Plus (tm) offers the most power and flexibility with the least programming work. The dBASE III Plus (tm) system also offers a menu driven program to build databases and create report programs compatible with the database structures. Additionally, dBASE III Plus (tm) has a wide base of commercial and free software that supports program development. The dBASE III Plus (tm) software is readily available through commercial outlets and General Services

Administration (GSA) purchase schedules. The capabilities of the dBASE III language can be extended by using a compiler and a screen design program (or code generator) as discussed below.

Compiler programs are software programs that convert a program file written as text into machine language or machine code, which the computer understands as instructions. Compiled programs do not require other programs, but can execute the machine code by itself. This 'stand alone' capability is useful because it removes the need for a second software program to run the original text code. A compiled dBASE III Plus (tm) text file can run without the dBASE III Plus (tm) program. A compiled program protects the source code of the program, since the compiled version is machine instructions, and bears little resemblance to the orginal text code. A user cannot make changes to the program and thereby cause loss of valuable and irreplaceable data. Also, a compiled program generally uses less disk space than a separate program and text file program. A compiled program will also execute faster when running, a benefit when organizing large amounts of data. Programs which are not compiled are interpreted or executed a line at a time. The interpreter views a single line of text code, converts it to machine language and executes the instructions. Interpreter driven programs are slow because the compiler does this for each line of text code.

Several compiler programs are available through commercial sources. The available programs are Clipper (tm), FoxBASE (tm), and QuickSilver (tm). All three programs offer an expanded version of the dBASE III Plus (tm) language that enhances the capabilities of dBASE III Plus (tm). All three programs will produce machine code. The deciding factors for selection were availability and price. Clipper (tm) was available through local sources, and programming help was available from the faculty. In addition, the Clipper (tm) program was available for a lower price than the other two programs.

One other software program was useful in developing the software for this study. The program is UI Programmer (tm), a code generation program. UI Programmer (tm) translates graphic designs into text program instructions. The graphics can be menu screens, input and display forms, or informational screens. UI Programmer (tm) creates text programs for dBASE III Plus (tm) and Clipper (tm), among others. The text programs created are compiled by Clipper (tm) into executable form. The UI Programmer is Object Oriented Program (OOP) software, which creats each screen design as an object and creates text code to relate that object to other objects in the screen. The OOP technique of program design is present in many commercial software programs. This technique is seen in many programs that allow the user to move program files, applications and windows around the computer screen. The UI Programmer (tm)

software is not necessary for the study, but is a useful tool, since it can reduce the time spent designing, writing, and trouble-shooting a program. The program was available through commercial sources.

Similar applications of dBASE III Plus (tm).

Several programs using dBASE (tm) are in use in the Air Force, the Department of Defense, and its' allies for managing personnel data, vehicle data, and supply information.

The first program identified is a product of a March 1987 Naval postgraduate School thesis. The system supports the Republic of Korea (R.O.K.) military personnel management system. The system uses dBASE III Plus (tm) software and data extracted from that nations' existing database management system (17:13).

The second program identified is a product of a 1986 Naval Post Graduate School thesis. The Student Mix Software System (SMSS) assigns students to seminars based on user selected rules, and to prepare the required output reports. The system also uses dBASE III as the software for creating the database and manipulating the data into reports (24:iii).

The third program identified was an AFIT thesis written that developed a WRM Vehicle Management program. The program is a dBASE III Plus (tm) system for maintaining information on WRM fleet vehicles (29:v). The software was

delivered to Air Force bases in the US, and is still in use by many transportation units.

The fourth report identified was an AFIT thesis written in September 1987 that created a DBMS to aid the U.S. Navy in producing high-priority 'Hot List' reports (25:xi). The system uses dBASE III Plus (tm), and the program meets or exceeds the requirements identified. Smith noted that the resulting software program was easy to install, easy to use, and did not require knowledge of the dBASE language by the user (25:xi).

Other applications of dBASE III Plus (tm) in civilian environments underscore its usefulness in managing database information. The use of dBASE III Plus (tm) in a telemarketing package gave it a distinct advantage over similar programs (26:57). Advantages cited were its speed, flexible reports, and form-letter generator. In addition, dBASE III Plus (tm) has received strong ratings by critics evaluating commercial software (19:121).

The successful use of dBASE III Plus (tm) by both the military and civilian companies for problem solving is testimony to its capabilities. It can support larger information systems, or be used by itself. The dBASE III Plus (tm) system can manage personnel data as well as transportation data with positive results.

Summary of Chapter II

The creation of a database requires an understanding of the basic terminology that describes a database, the

characteristics of well-made databases, the objectives of designing a database, and selection of software to support the database design. Last, the examination of similar program applications shows that the software selected will support the thesis.

First, the basic terms are data item, record, file, and database. A database management system is the collection of these elements that serves some common purpose or enterprise. A database can take one of several forms: hierarchical, network, or relational. The difference in the three forms is how the nodes within the structure link to each other in parent-child relationships.

Second, the characteristics of a good database are data independence, data redundancy, and data integrity.

Third, the objectives of designing a database are to support several functions related to the enterprise, while preserving data independence and integrity. Minimizing data redundancy ensures the program uses the smallest amount of disk space and computer memory.

Finally, there are similar programs which confirm the use of dBASE III Plus (tm) as a database management system for effective management of large amounts of data.

III. Methodology

Introduction

This chapter outlines the methodology used to develop the database management system application. The ten steps include the collection of user requirements, verify user requirements or information, derive minimal cover, conversion to a static, logical database, augment the static database, convert it to a physical database, estimate minimum response times, and optimize user requirements. The last step of the methodology ends with a flexible, recoverable, user friendly database application.

Program Design Cycle

Vesely outlines a design cycle for development of a database system as having ten steps, as shown in Figure 1 (30:2).

This design assumes that someone accepts responsibility for the program after delivery to the user. Program errors may require additions or modifications, which should be done by a qualified programmer. A large business or firm which decides to invest in a database system will have a programmer or computer section capable of performing the required program maintenance. The researcher delivered the prototype to the Standard Systems Center at Gunter AFB, but the program lacks a sponsor to bear responsibility for

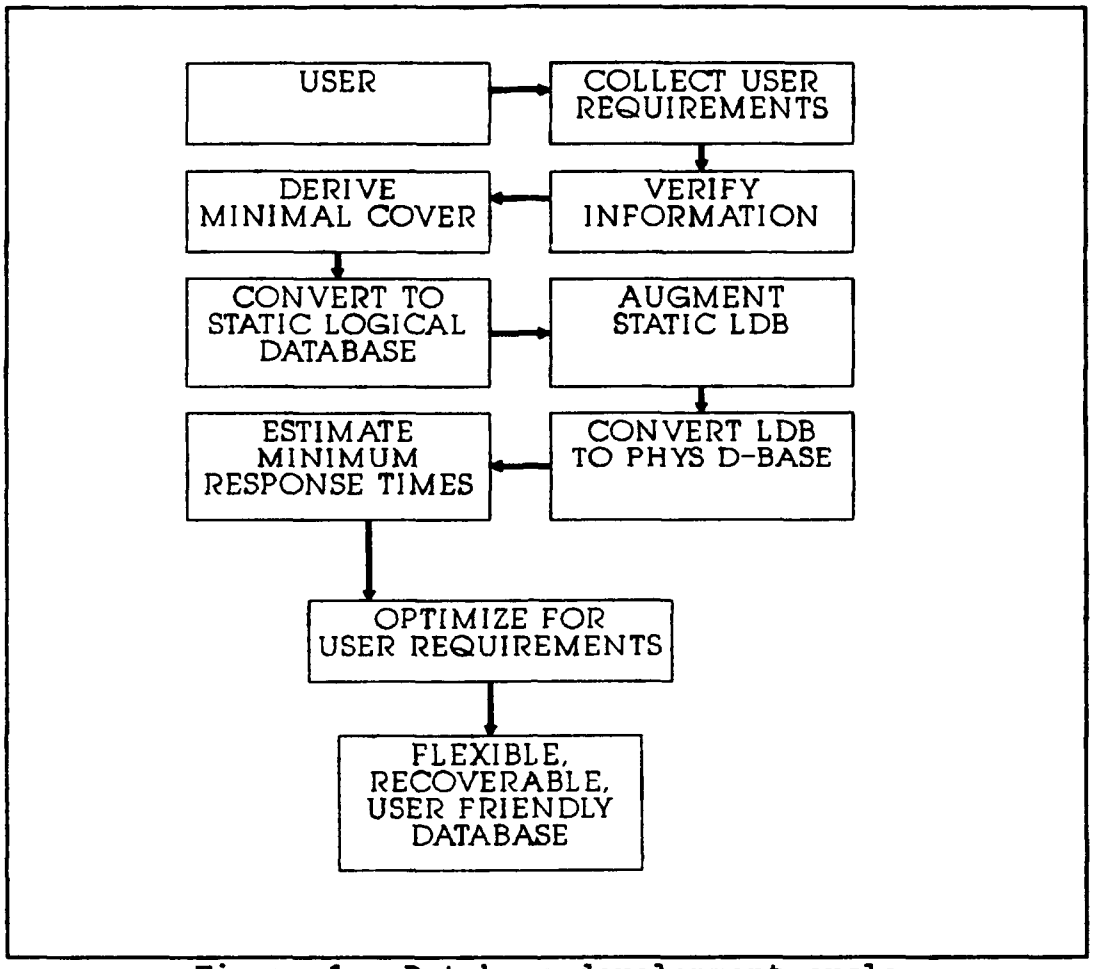


Figure 1. Database development cycle

further development, distribution, and maintenance of the program. In a program developed for a business, a poor or useless program is not used if it fails to provide the product or advantage needed. Similarly, a program is not used in the Air Force if it does not provide some advantage or increased benefit. This prototype program requires a validation of the program to decide if a software system would be helpful in managing a WRM program.

Collect User Requirements. The development of WRM software will use a model designed from requirements outlined by experts and functional WRM managers. First, the researcher conducted telephone interviews with experts at the Logistics Management Center (AFLMC/LGXW), at five of the Major Commands, and with at least one experienced functional managers of WRM assets at the base level within each Major Command. The researcher solicited five of the Major Commands for inputs: the Strategic Air Command (SAC), Military Air Command (MAC), Tactical Air Command (TAC), US Air Forces Europe (USAFE), and the Pacific Air Forces (PACAF). The Major Command experts were from the Logistics Plans and Programs office, the office of primary responsibility for administering policy and practice of the WRM program. The researcher interviewed one functional manager from a base in each command, selected by recommendation from the Major Command offices. There were eleven initial interviews, with other interviews conducted with the users to clarify previous inputs and allow expansion on the user's ideas. The evaluation also included five more managers, one from each major command, to get a larger sample from which to draw a consensus. The result was sixteen possible evaluations. No statistical analysis of the evaluations was performed.

Several iterations of the interview process developed a clearer understanding of program requirements. There were some requirements beyond the capability of the researcher to

address, due to complexity or software limitations. The requirements which were not addressed by the researcher are included in Chapter IV. The unmet requirements were not critical to the research, and could be met with additional research, and a dedicated programming effort. Establishing requirements among the experts and managers was the most difficult part of the requirements determination.

The researcher selected telephone interviews for determining requirements for several reasons. Interviews provide feedback quickly, allow flexibility and can be repeated several times to expand ideas. Telephone interviews can be done quickly, the structure of interviews allows exploration of ideas, and clarification of confusing ideas (13:169). Several telephone interviews with each expert developed ideas into clear requirements.

The individuals interviewed were not experts in computers or program development, they are actual WRM program managers or past program managers who have developed expertise dealing with the WRM program. They should therefore be representative of functional WRM managers.

Verification of User Information. The different data elements needed for the program came from the available Base Supply reports and interview results. Obtaining the data from the Standard Base Supply System (SBSS) required a specialized computer program to extract data from the SBSS in a useable form. The individual users supplied some data elements, as they are unique to the base or WRM program.

Second interviews verified the need for major program elements and functions. In the third interview, the author confirmed the data elements recommended or desired. Some data elements came from other sources; for example, the unit price and total quantities determine the value of inventory on hand.

Derive Minimal Cover. The identified data elements were separated into related or natural groups of information. For example, on-hand inventory balances, authorized quantity, and authorized unsupportable quantity were grouped as inventory data. By grouping information, it became easier to define the relationships between groups of data. A grouped data set will have a 'key' field which uniquely identifies the data record. This process, called normalization, groups data under a key field. The key field provides each record a unique identity so the record can be moved, rewritten, or deleted as a group. Normalization of the data sets then eliminates redundant data and provide a logical method for locating unknown data by related known data (30:5).

Convert to Static Logical Database. Another consideration was the amount of data to store in each data set, and as a whole in the system. An extremely large data set might need to be broken into two or three separate data sets to avoid having too large a data set for the system to handle. Also, adding information to the system makes the

data set grow. The design allows for the added information so the system does not exceed physical limitations and thereby 'crash'. The anticipated number of inserts, updates, and deletes made to the data set were also considered (30: 6). Any insertion of a new record, overwrite, or deletion of an existing record may require additional actions by the system to maintain file size. A deletion, for example, would require a repacking of all records with a sort of the records, and a re-creation of the associated index file. This process takes time, and the system should minimize the time spent by the user waiting for the system to complete required actions. The design process anticipates number of queries using second keys, described in the next step, as these queries could also impact the system's 'dead' time. Dead time is the time that a computer is searching for data or performing some other task when the user cannot make inputs or perform other computer operations. In effect, until completing the query, nothing else can be done with the computer.

Augment Static Logical Database. The next step requires identification of the primary keys and secondary keys, as well as foreign keys. A primary key is a field within the table that points to one and only one record of the table, a primary key is the way to find a unique record. A secondary key is useful when there is some information about the desired record, but not enough to uniquely identify it. A secondary key would locate many records that

match the desired data. An example of a primary key is the Social Security Number, which uniquely identifies every taxpaying citizen. A secondary key could be the last name, which would narrow the search considerably for a particular individual; the name 'Smith' for example, could belong to many records.

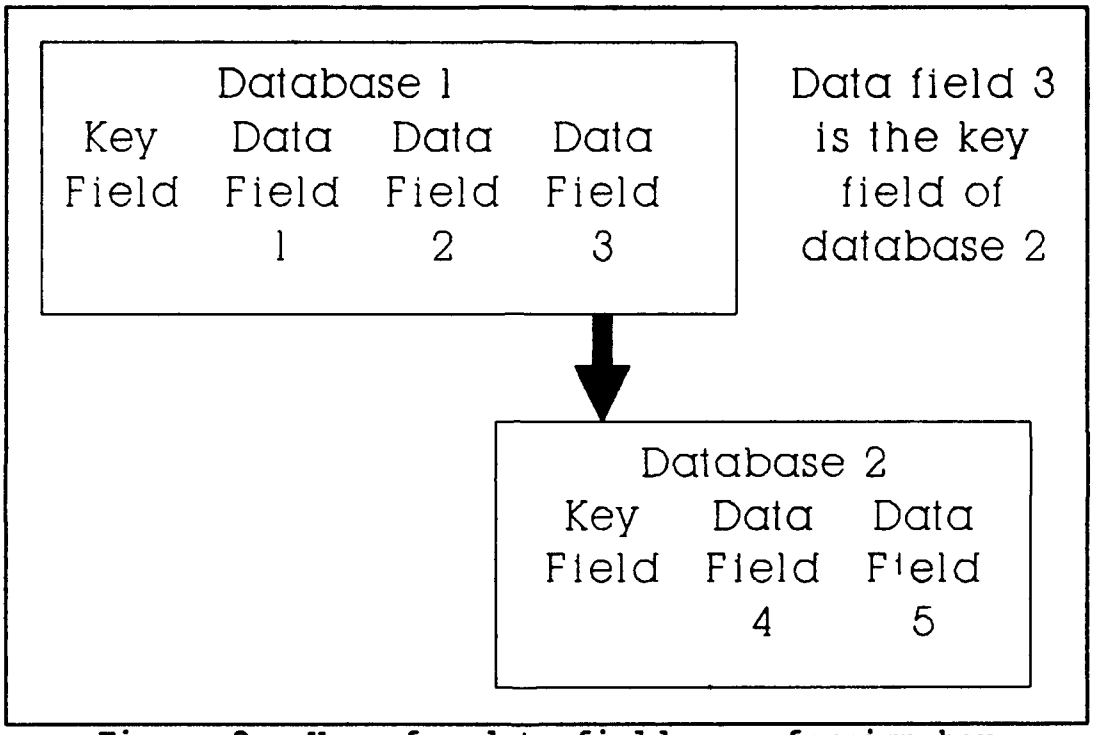


Figure 2. Use of a data field as a foreign key

Some data tables use a 'foreign key' to connect one data set to another, as shown in Figure 2. An example of a foreign key is a data field within a table that relates to the key field of a second table (30:119). An item discrepancy table would contain a field for the organizational number used as a key field in a second table to find the organizations name, phone number and point of contact.

Conversion to Physical Database. The translation of the users requirements into program design was made easier through the use of a database software development system. This software, UI Programmer (tm), is a database system tool that decreases the amount of effort and time required to develop a system. UI Programmer (tm) generates dBASE III Plus (tm) and Clipper (tm) program code for menus, reports, sorting routines, organizing, and relating database information. The dBASE code is compiled using Clipper (tm), a dBASE compiler, to produce an executable program. Using UI Programmer (tm) to design input and output screens, instructions for how to process database information were attached to these screens. UI Programmer (tm) used these screens and instructions to write the Clipper (tm) program code. The result of this process was a physical database program that meets the program requirements outlined by the users.

Estimate Minimum Response Times.

Database management system applications can suffer from long delays in processing new data, recalling old data, sorting records, and presenting reports created from the data. The design of the database management system application had to consider the number and kinds of search operations the system had to perform to meet user requirements. For example, searching for a specific

document number in the database can use a general search of the database or a search of an index file.

Index files are separate files containing the key field of a database and the location of a corresponding record in the database. An application can quickly search the index file and then jump directly to the appropriate record. Alternatively, an application can directly search the database key field to locate a record. A search using a index file is a much faster search method for a database with many records, but not for a database with few records. The difference in the speed is a result of the amount of data the computer must store in its memory to compare against a desired search value. For a normal search, without an index file, the computer stores all the data from a single record in memory, but compares only the key field against the search value. Moving the entire record data in and out of memory requires more time than a single, indexed field value compared to the search value.

This application has the potential to maintain records on thousands of supply documents, and even greater numbers of WRM authorizations. These databases required index files to ensure minimum waiting times for the user.

Optimizing for User Requirements. The first version of the program, Version 1.0, was tested by one of the WRM managers. The author worked with the manager to test the capabilities of the program in the field using procedures outlined in the User's Guide. The manager (or tester) tried

every function of the program. It was assumed that major program errors were identified at the time. Existing problems or difficulties were noted and corrected in the next version of the program (14).

The user who evaluated the initial software was familiar with WRM and computers. This enabled the manager and researcher to readily exchange ideas about the software. The manager who evaluated the initial program was an experienced WRM manager, and is responsible for some WRM software at the Air Force Logistics Management Center. Testing of the Version 1.0 by a functional manager ensured the program was moving toward meeting the objective of the program development. It also ensured the program worked in a logical manner in doing the needed functions. This step represents the ninth of Vesley's ten steps. Having developed the application, the last step results in a flexible, recoverable, user-friendly database.

Version 2.0 Development. From feedback from the on-site test, a revised version of the program, called Version 2.0, was developed. The second version included corrections in program execution and omissions in the User's Guide. Once these changes to the program were completed, the program was mailed to the users for evaluation.

Program Validation

In the last step, the managers evaluated the final version of the program. Evaluation was performed to ensure

that the program meets the requirements outlined by the experts and managers in the interview process. The program, Users Guide, cover letter, and response card were sent by mail to the users for evaluation. Evaluation by mail does not allow for problem solving of software problems or exposition of instructions, but is a cost effective method of having many users try the program. By having the program evaluated by the same users that developed the requirements, it was easy to determine if the program is helpful. The managers decided if the program meets the objective of the study. Several users also provided feedback about the program and recommendations for changes. This final version and User's Guide, along with any open additional requirements, was delivered to the Air Force Logistic Management Center at Gunter AFB for distribution and possible further program development.

Summary of Chapter III

The process of designing a database management system application used a methodology having ten steps. These are: collect user requirements, verify user requirements or information, derive minimal cover, conversion to a static, logical database, augment the static database, convert it to a physical database, estimate minimum response times, optimize user requirements, and end with a flexible, recoverable, user friendly database application.

The key element of the design cycle was determining and verifying user requirements. The requirements determined

the initial capabilities and functions of the application. If the requirements do not address the needs of the users, the resulting application will not meet the needs. Each additional step of the design cycle contributed to the flexibility, speed, and usefulness of the application. The application was optimized by having the application evaluated and reviewed by a user experienced in both WRM management and computer applications. Errors were corrected, and the application was evaluated by many users to determine the usefulness of the application.

IV. Findings and Discussion

Introduction

This chapter outlines the program developed from the previously discussed methodolgy. The user requirements formed the authorization, supply, inspection, and quantity functions. Other functions, such as the reports and listings function, were developed to support the basic program functions.

User Requirements

User requirements were collected through telephone interviews. Some of the requirements involved simplification of existing reports, such as the War Consumable Distribution Order (WCDO)/Base Level Support Spares (BLSS) R07 Report, while other requirements were an amalgam of existing data. The users identified many requirements for a WRM program, several of which were beyond the scope of the research or the capabilities of the researcher to develop.

The inputs from the users determined the major program functions, as shown in the program main menu in Figure 3. User requirements became the basis for the basic authorization, supply document, inspection, and quantity balances functions. The functions for organizational point of contact, supply details, reports, reading in, and backing-up database information were developed to support

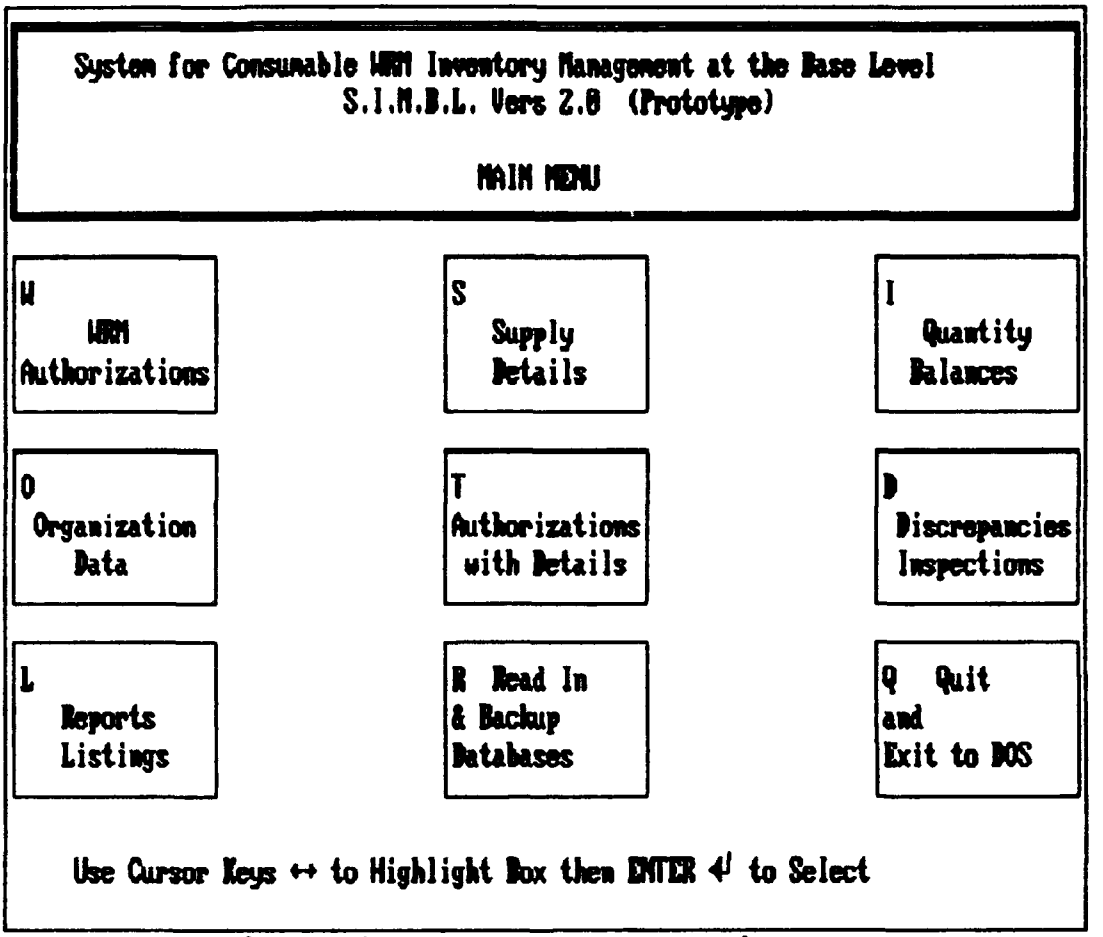


Figure 3. SIMBL program main menu

the basic functions. From this menu, the user can select any of the program functions.

A common need of the users was a technique for handling WRM authorizations, supply document data, inventory balances, and surveillance visit or inspection reports. Also, the users wanted to print listings of the data, along with supporting information. For example, at least one of the users wanted to link a WRM authorization to all the supply documents related to the authorization, with an option to print the listing (9).

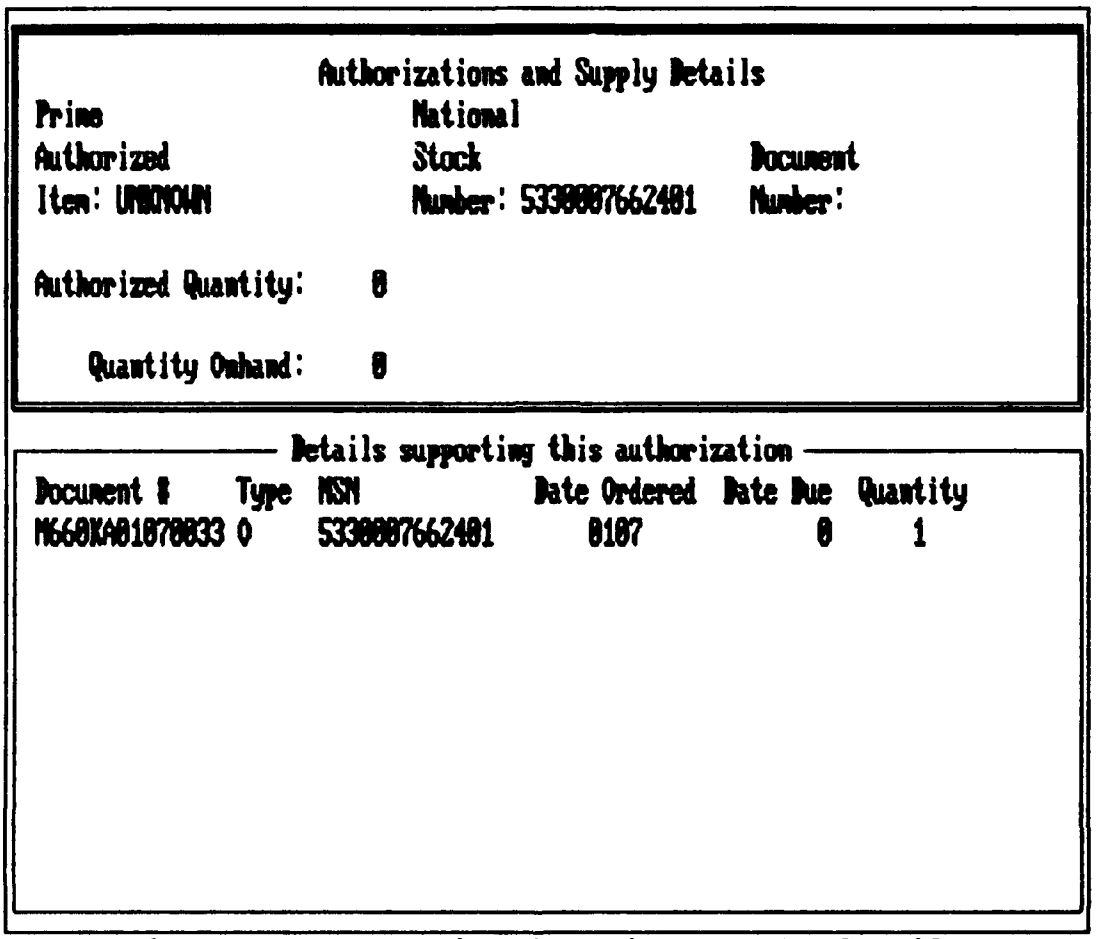


Figure 4. Authorization with supply details

Several of the interview subjects expressed a desire to tie supply information to unfilled WRM authorizations. In some cases, an authorization exists, but the item is on order and awaiting funds (memo due-out) before it can be bought. Some high cost equipment falls into this category. Some consumable items may not be available, or are due-in from the source of supply. The researcher addressed this need by creating a program option relating an authorization to the supply details (14). This function allows the user to move through the database of national stock numbers while displaying the descriptive phrase for that stock number.

Figure 4 illustrates this function: the top window shows the authorized stock number and the bottom window shows the supply details. When the user finds the national stock number of interest, the program displays a list of all supply documents associated with that stock number.

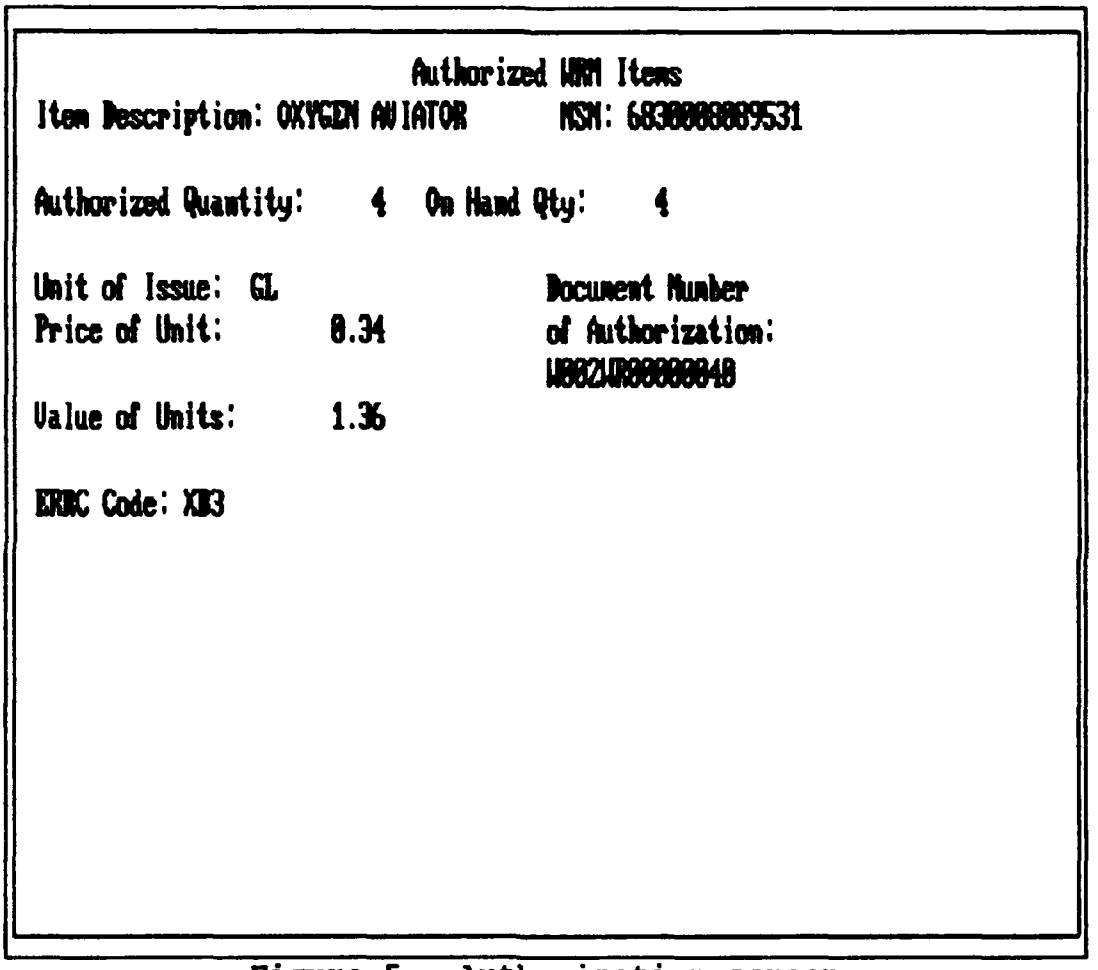


Figure 5. Authorization screen

Each authorization requires a unique national stock number (NSN), which also appears on supply documents. The stock number is an example of a foreign key as previously discussed, which provides the key to locating supply documents. Since each authorized NSN may have several

documents related to it, a second database of supply documents was needed. The second database used was the supply document database, which uses the NSN as a secondary key field. The primary key for the supply document database is the document number, which uniquely identifies each document.

Relating authorizations and supply documents assumes this information is available. This necessitated two program functions, one for authorizations and another for supply details. The authorization screen shown in Figure 5 presents basic information about an authorization. It allows the user to view pertinent data about the authorization.

The second required function deals with supply documents. The supply documents function, pictured in Figure 6, presents basic information about each supply document. Supply documents created with this function show up in the program function relating authorizations with supporting supply documents.

Most logistics planners deal with many organizations, each of which may have several individuals responsible for WRM items. For example, a supply squadron will have separate points of contact for gun oil, engine oil, and aviation fuels. A method is needed for referencing the point of contact for each WRM item or organizational shop (20).

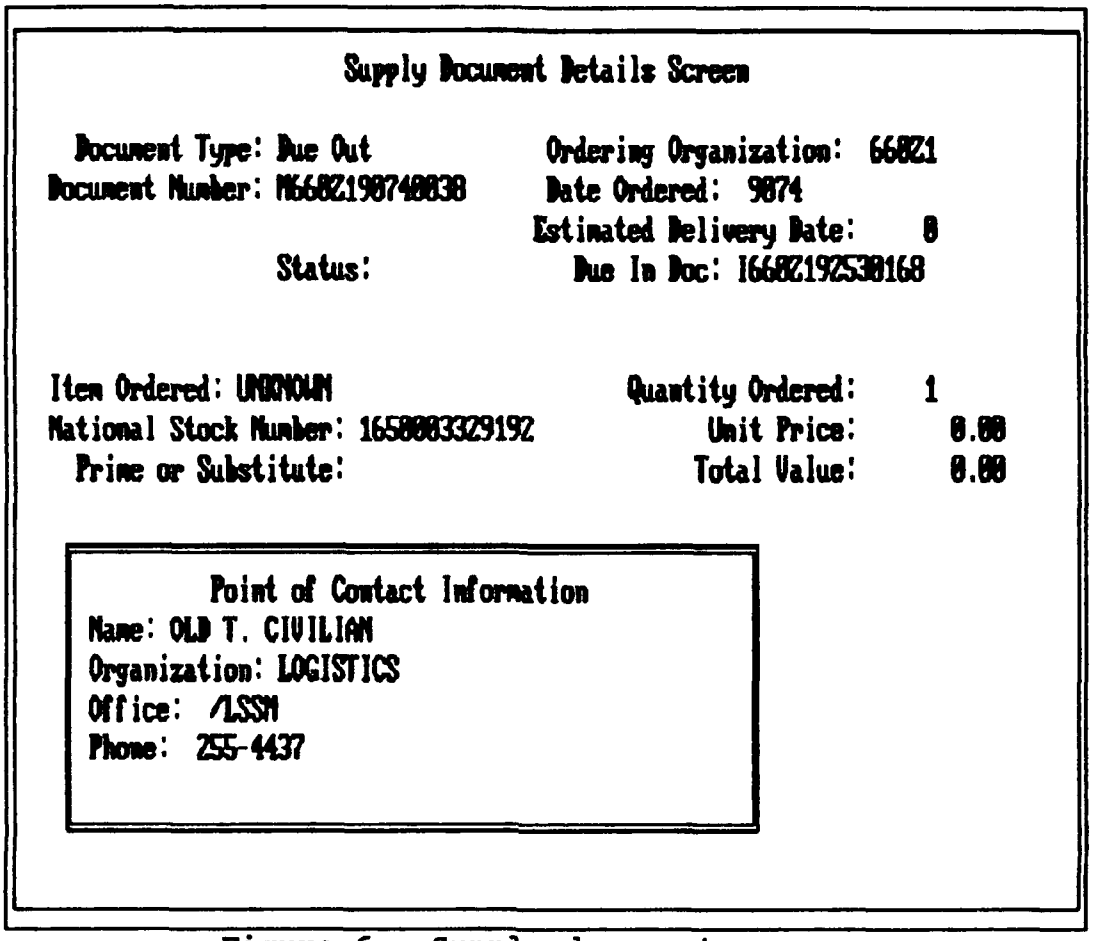


Figure 6. Supply documents screen

The point of contact for each supply document is shown in the lower left part of the screen in Figure 6. The point of contact data is entered into the computer in a separate function. To fulfill this need, the researcher created a database of point of contact (POC) data where the organizational shop code is the primary key field. The organizational data screen, shown in Figure 7, allows the user to build a library of information on the points of contact for each organization. Some of this data appears in the supply documents function, a report of point of contact

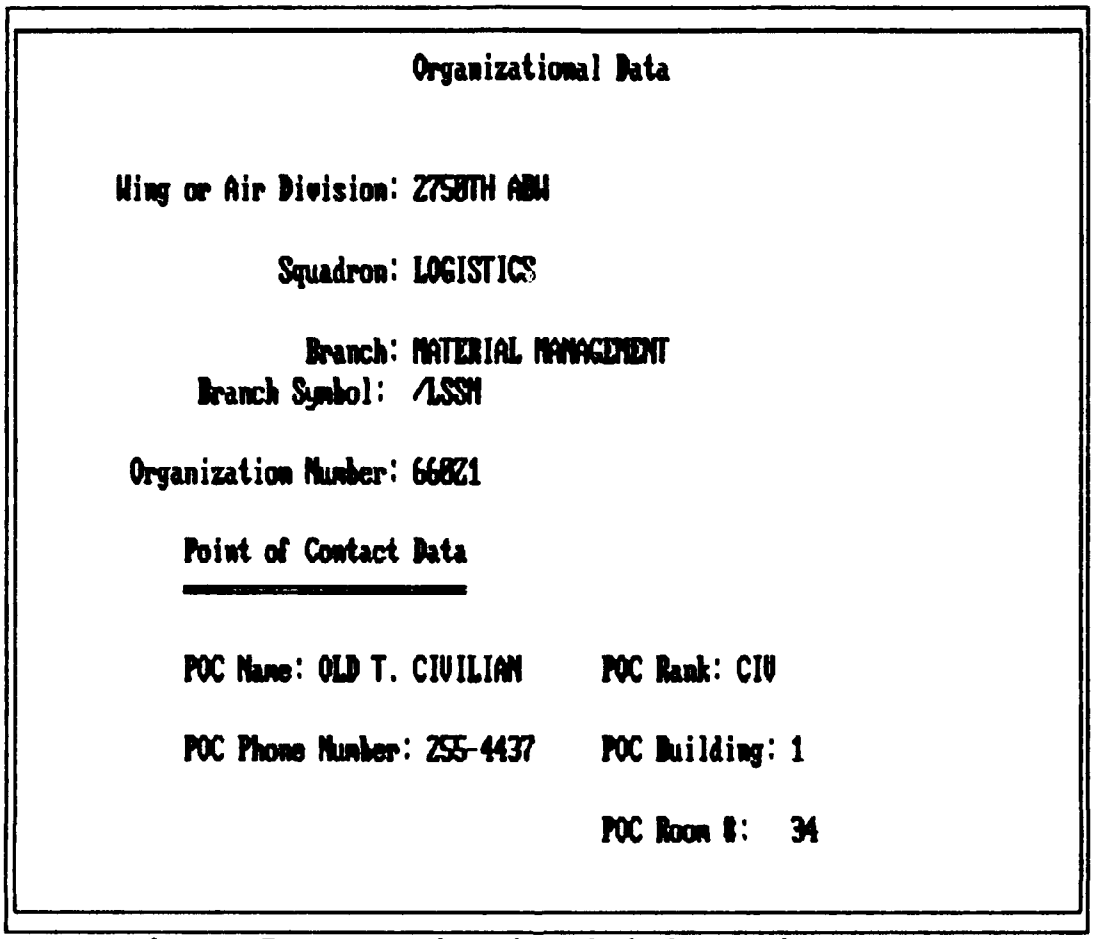


Figure 7. Organizational information screen

data, and in a report of inspection data.

Inspections of WRM items involve a significant amount of time for the base logistician. Some of these inspections result in deficiency reports and follow-up inspections. In addition, these inspections occur on a semi-annual basis. A method to track inspection dates, open and closed inspection reports, discrepancies, and responses is useful.

The program meets this need by creating a database of inspection data, keyed to the organizational shop code. Figure 8 shows the screen used to enter and view inspection results. This database is referenced by the organization

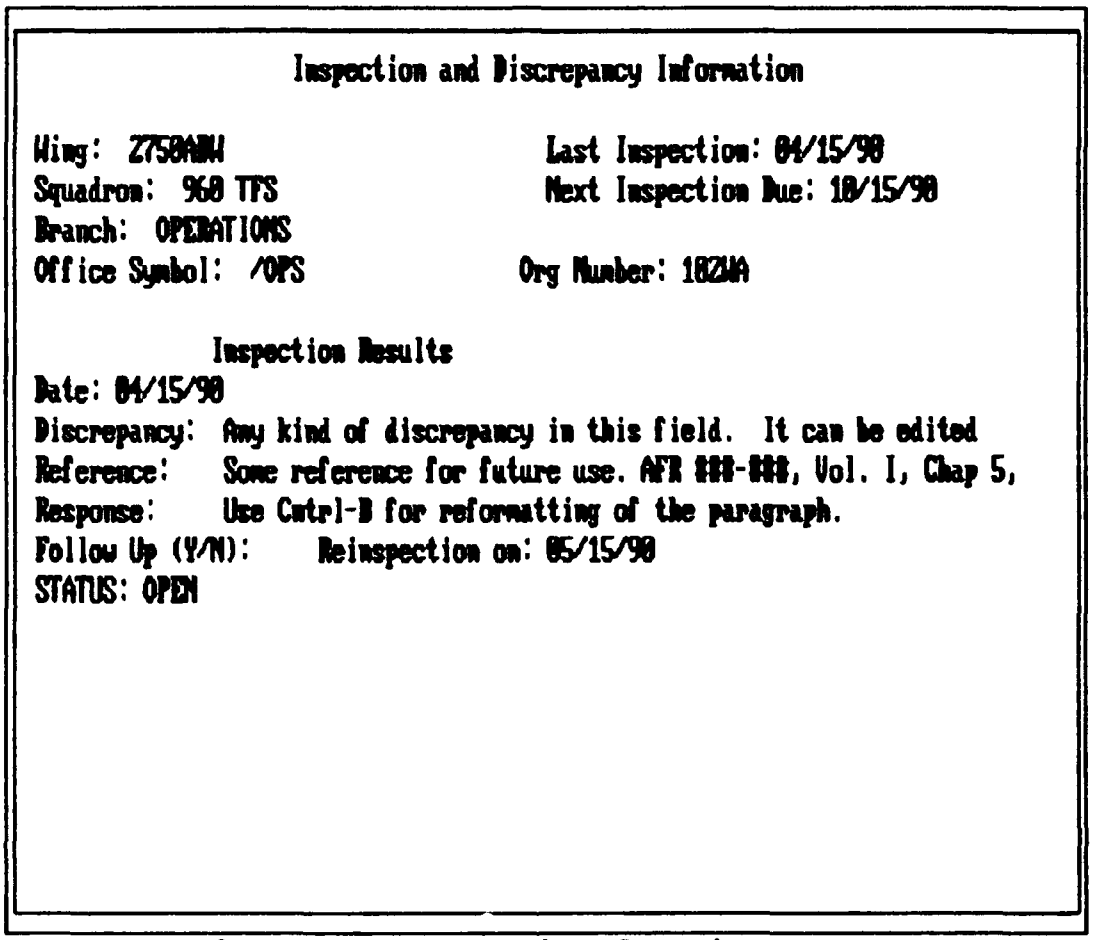


Figure 8. Inspection function screen

shop code number, which is entered from the keyboard using the search option. Inspection dates recur on at least an annual basis, and a method of tracking inspections, annotating discrepancy information and scheduling subsequent inspections is available in this function. The program function addresses this need by maintaining the results of WRM surveillance visits and inspections through memos, as discussed below.

The inspection program allows the user to record inspection data, results, and to schedule reinspection dates. Every inspection can have three memos associated

with it. Figure 9 is a close-up example of a memo box that appears in the bottom portions of the inspection screen. Some inspections will need only short memos, while other memos are larger; the program allows the user to enter memos as long as 64,000 letters.

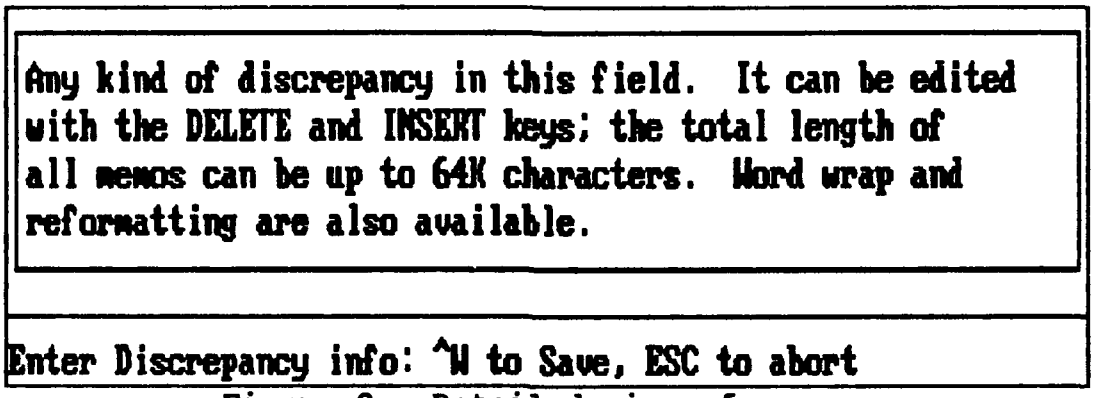


Figure 9. Detailed view of a memo

Reports and Listings Requirements.

Each of the logisticians had ideas of what reports were needed, which items to include, additional data to present, and what format to use in the reports. The researcher looked for common elements among the requirements and developed listings to support the needs of the users and the major program functions. The result was the menu of reports listed in Figure 10.

There was a need for a listing identifying all authorized/on-hand imbalances for every item tracked as WRM. A combined listing is more useful than the three individual listings now available (9;7). This listing could also include a requirement for a munitions listing identified by

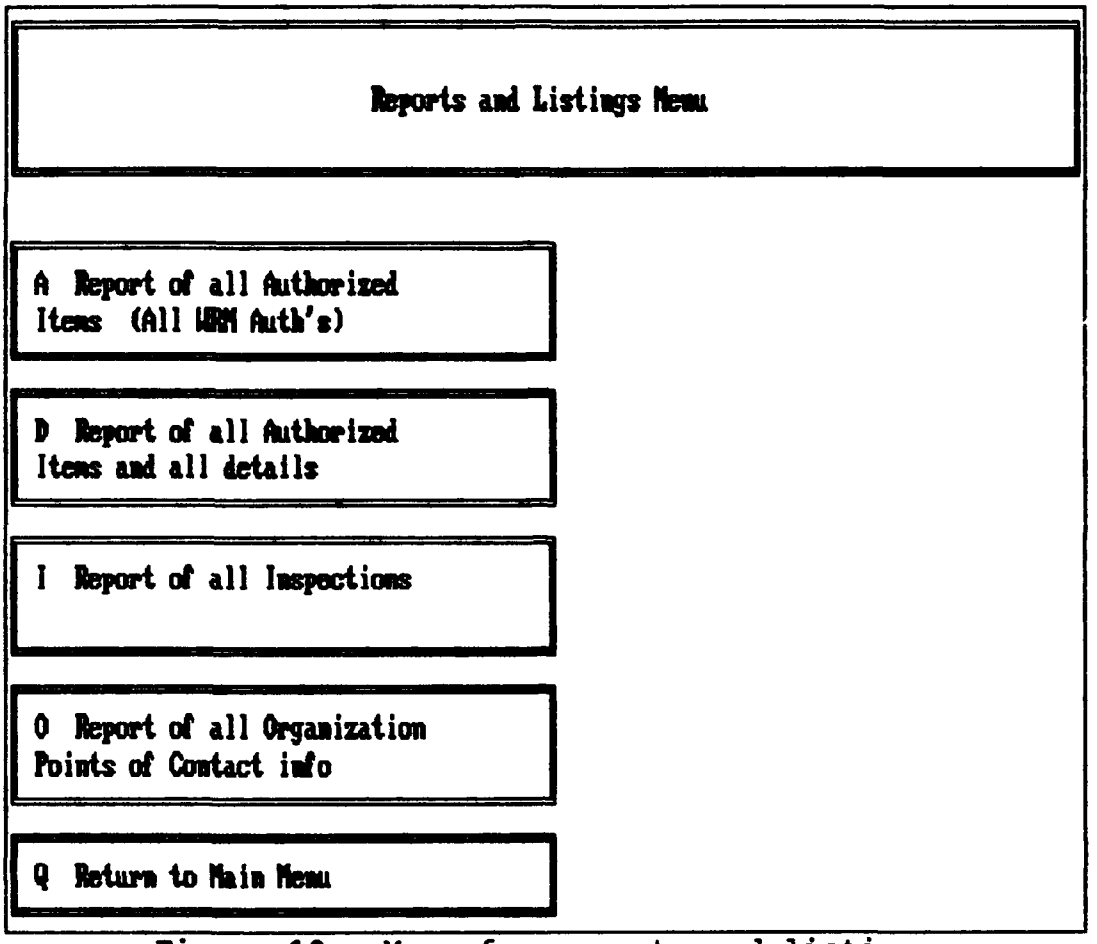


Figure 10. Menu for reports and listings

Chief Master Sergeant (CMSgt) Stock. Munitions were not included in the program, as there is a program for munitions under development (14;27).

Differences between quantity authorized versus quantity on-hand are of interest, and several additional pieces of information should be presented: due-in and due-out document numbers. This allows the logistician to identify those items to look at more closely (20). The report and listings menu has an option to generate a listing for each authorization, with the current inventory balance and any supply documents.

A listing of which units and areas need inspection would be useful (7;20). There is an option on the reports and listings function that generates the appropriate report.

A large WRM program has many points of contact the logistician needs to know, particularly when the unit is inspected infrequently (20). For example, a unit may have several pieces of equipment, each being managed by a different shop and point of contact. A tactical ground control unit has radar equipment, vehicles, and munitions, each having a different point of contact. The report of organization points of contact provides a medium for the base logistician to track the responsibility in a multitude of organizations for each shop code.

All the base-level logisticians interviewed stated that the present R07 WCDO/BLSS report provides more information than they use on a daily basis. CMSgt Stock of the SAC/LGX office expressed the greatest concern with authorized versus on-hand quantities (27). Imbalances in these inventory levels are of high interest at both the base and Major Command offices.

There is a need for a Vehicle Status report to reflect storage costs, maintenance costs, and the vehicle status (9). A report of this data is available through the Air Force WRM Vehicle Management System, although the system does not provide cost Figures for the vehicles (29:62). Because of the complexity of obtaining the appropriate data

and updating the database, the researcher declined to pursue the requirement.

The possession of pallets and nets change hands rapidly with deployments, as well as their location and condition. A simple way to annotate the transfer of pallets and nets from one unit to another would be useful (27;20;22). The scope of the program encompasses consumable items but pallets and nets are equipment, and these items are not part of the program.

The ability to produce a listing by storage location would be useful, when the materiels are in storage at several locations for the organization (8). Bases in the Pacific Air Forces (PACAF) and the United States Air Forces in Europe (USAFE) must keep track of items stored at multiple locations. Some overseas bases are responsible for WRM equipment stored at dispersal points. When the location code, a four letter identifier, is linked with the name of the location, the information is classified SECRET. This program is not designed to process classified information, so location codes and location names are not included. A later revision of the program may include the four letter codes.

The money spent on a WRM item is of interest to the base logistician. The money comes from a WRM budget controlled by the base resource manager (8;7). Dollar expenditures are not in this program, as it would greatly complicate the programming, and involve accessing accounting

and finance records in addition to base supply records. Collecting dollar data from Accounting and Finance records would greatly complicate the program, and require continuous updates of those data elements.

Interface with the SBSS

An interface with the SBSS would be useful to allow updates on the status of WRM item due-in and due-outs (7). The database management menu in Figure 11 shows the options available for reading in SBSS extracted data, backing-up the current databases and restoring previous databases. The first menu option uses a data file created by a separate program which is run on the SBSS computer. This separate Query Language Program (QLP) file was developed by a technical mainframe supply programmer at the request of the researcher (2). The QLP pulls data from the SBSS, sorts and rearranges the data into a temporary file, which is transferred to a computer diskette. The file is read by the SIMBL program, and the data is stored in the either the authorization or supply document database.

The two last menu options were necessary in case of a failure in the system, as discussed earlier. The process of backing-up and restoring database files is simple and easy. Backing-up the database files is a much easier process than reading the data into the computer. The user must place a computer diskette, also called a floppy disk, in the disk drive, and select the menu option. The program will copy the database files to the floppy disk. If the copy was not

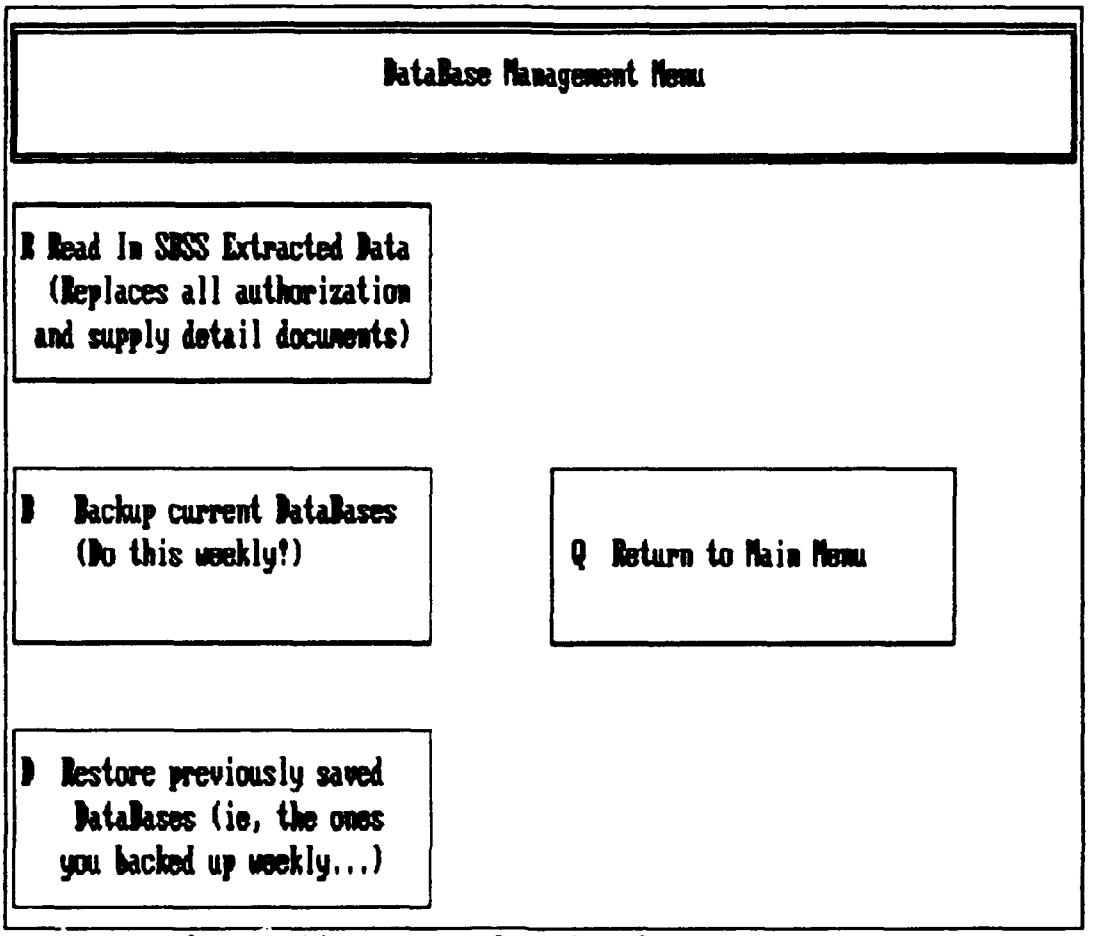


Figure 11. Menu for database management

successful, the program tells the user of the problem.

The last menu option recovers the computer database files from a floppy disk, and returns the program to the current status it was in when the files were last saved.

Incorporating Good Database Design

The researcher ensured the program included good database design characteristics in several ways. The program provides data integrity by giving the user a method of backing up all the database files. The program also restores the database files from the user's backup disk.

Data redundancy is minimized in each database. Some data is stored in separate databases that allow access from any portion of the program. For example, the name of a point of contact is used in the supply documents function, the organizational information function, and in several reports. Having a single table of this data eliminates the need to store the data at least three times for three functions.

Version 1.0 Evaluation

The first version of the program was examined by Lieutenant Colonel Campbell and Major Steve Hagel of the Air Force Logistics Management Center. They made many recommendations for the program about objectives, program design, program functions, screen appearance and the report listings (14).

Program Objectives. The evaluators recommended the program objective to include just consumable items, excluding War Readiness Spares Kit (WRSK) items. They stated there are several tools available for handling other WRM information, but there is no tool for just consumable data. Also, for exchanging information from the SBSS to this personal computer (PC) software, they recommended the establishment of an interface file to maintain compatibility with future versions of the SBSS and the PC program. This suggestion will be a recommendation for a fully developed version of the program.

Program Functions. The LMC evaluators recommended several changes in program functions. The first suggestion was to display all in-use details that support an authorization. This recommendation is addressed by the supply details function.

The second suggestion was that when going from different sub-menus, the program remember the last organizational shop code accessed and locate the database pointer to the appropriate record in the next database. In effect, the program 'remembers' what organization you are dealing with, and looks for that organization when switching to another database.

The evaluators recommended a quantity balances function to identify overages or shortages and a comprehensive report list of all balances. This idea has merit, but could not be fully implemented. The data needed to calculate overages or shortage, such as the authorized unsupportable quantity is included in the quantity balances screen. Limiting the function to strictly those items with imbalances was not possible because it would be difficult to edit records where there is no imbalance between the on-hand and authorized quantities. This is a limitation of programming expertise, not an inherent problem with the data or the software. The current program design allows additions of data, but does not discriminate between records with imbalances and those with no imbalance. The function presents information needed to calculate the net stock position for each national stock

number. Figure 12 is a sample screen from the quantity balances function.

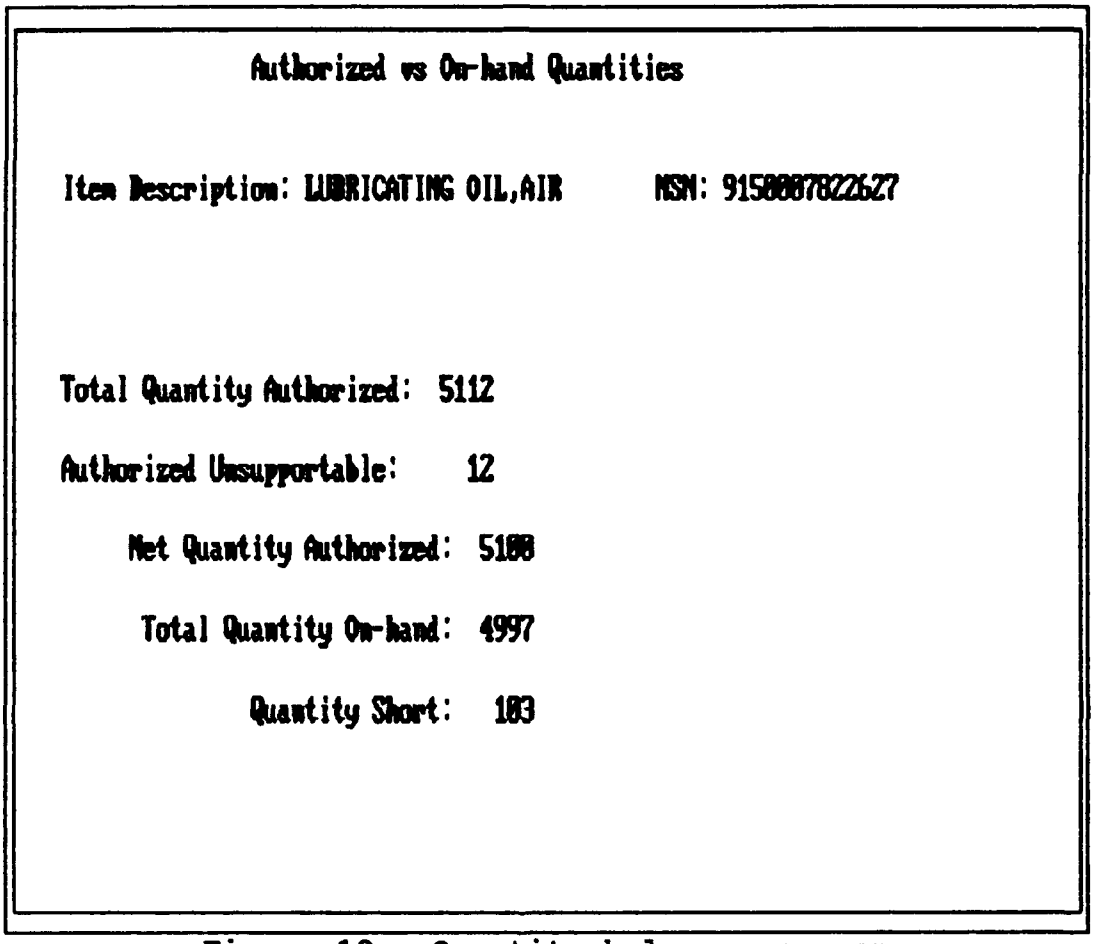


Figure 12. Quantity balances screen

Another recommendation was that a separate report for nets and pallets is probably not necessary, as the users who manage these items are those heavily involved in exercises and deployments. The status and location of pallets and nets changes rapidly. Just keeping up with the changes to this data would be difficult to do without developing a much more complex software program. Also, the information is not necessary in a program that deals only with consumable items.

Another recommendation was the addition of a memo field to the authorization screen for the user to make notes about an authorization, and a field giving the total quantity on-hand for each authorization. This last recommendation has merit, but was not included in Version 2.0 due to the lack of available time for program development.

Screen Appearance. The evaluators also suggested several additions to the screens. The first addition was a disclaimer on the program to inform the user not to input classified information. The disclaimer appears as a 'pop-up' box to remind the user before appending data records.

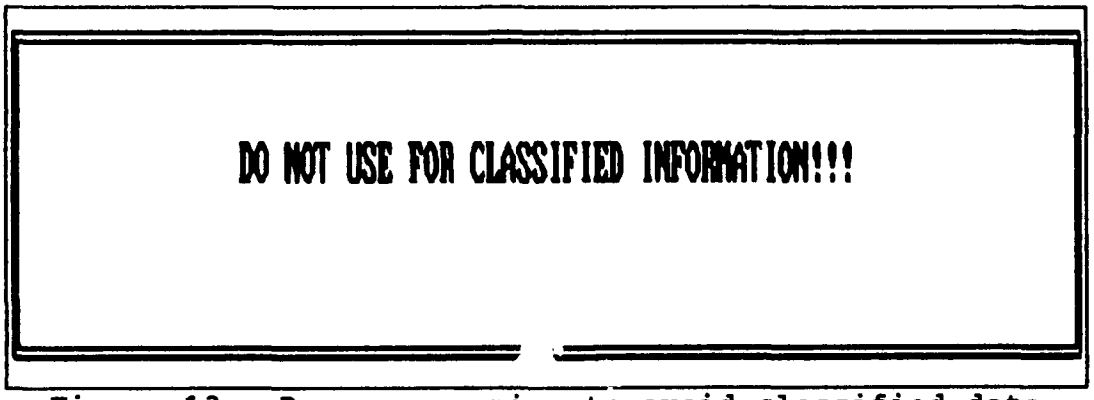


Figure 13. Program warning to avoid classified data

The program places the reminder on the screen as shown in Figure 13 before going to the main menu, and before appending new authorization records.

They also recommended a test of the program's screen colors on a Liquid Crystal Screen (LCD) and a monochrome monitor to ensure that screens are legible on common computer systems found in the Air Force. The colors selected for the final version of the program are clearly

seen on a LCD screen, but several users complained that the colors were garish on a normal color monitor (14;7).

Additionally, they suggested changes to several menu and screen texts to improve user understanding. One suggested screen change was to give the user a final opportunity to undo all deletions before exiting the program, avoiding permanent removal of all records marked for deletion.

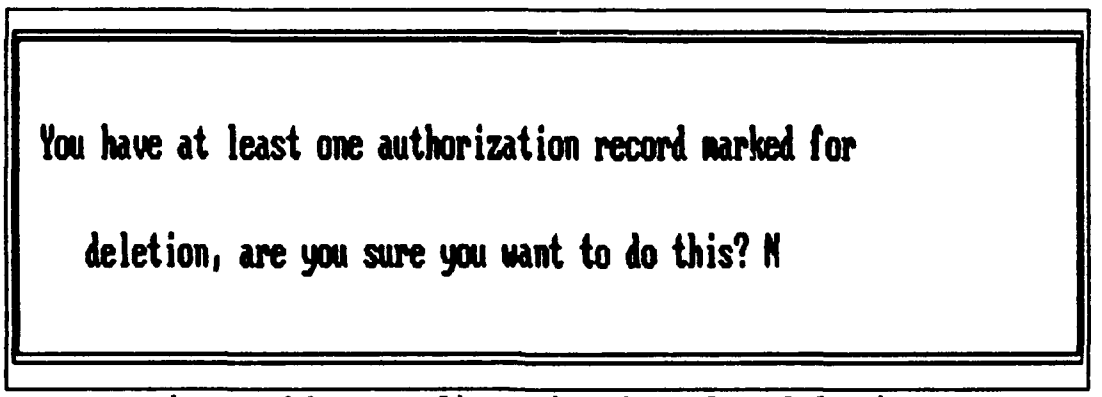


Figure 14. Confirmation box for deletions

This recommendation is included as a precaution against losing valuable information. An example of this confirmation 'box' is seen in Figure 14.

Reports and Listings. The evaluators recommended the listings of inspections due have a limit of 30 days from the current date or a user entered number of days. The evaluators stated the inspection schedule would change too frequently beyond a 30 day outlook. The report generated has a set length of 30 days.

They suggested a report be created that lists each authorization accompanied by the supporting in-use details,

due-out or DIFM details, and total quantities authorized, supported and on-hand. The reports and listings option of all authorized items and all details provides this data.

The evaluators also recommended that any listings to the screen pause for each full screen. All but one of the reports is created with the Clipper (tm) Report generator and will automatically pause after each screen. The listing of all available information for all WRM items is compiled into the program, and does not pause. This program flaw needs correction in a subsequent version.

Program Validation

The program was sent to seventeen logistic plans and program offices, five at the Major Commands, eleven at base-level offices, and the Logistics Management Center. The program, User's Guide, cover letter, and response card was sent to each office asking the users to install the program and databases and then to examine the program functions. Responses were returned using pre-addressed, stamped postcards. The postcard was selected as a method of response to make it as easy as possible for the user to respond. The postcards have a professional appearance with laser printing, were postage paid, and the postcards were readily available. The evaluators were asked to decide if the program would be useful in the management of their WRM program. Several users elected to respond in writing with their views, comments, and observations on the software in separate letters. Although not part of the scope of this

study, their recommendations are included in Chapter V for further program development.

Investigative Questions Answered

To develop an effective software program to track and manage WRM asset information, several investigative questions were posed.

First, can a program be developed that will meet requirements identified by logisticians and by Air Force Regulations? There are several programs written in dBASE designed to support Air Force operations, such as the WRM Vehicle Management System. The requirements for the prototype software were similar to the vehicle management software, giving credence to the concept of a WRM inventory management software program. By limiting the scope of the prototype software to unclassified, consumable items, the software met the requirements of users and Air Force regulations.

Second, what are the functional requirements of a WRM software system for effective collection, organization, and presentation of information? Data was collected automatically in the SBSS computer, removing the need for the user to input each piece of data. The needed data was available from the supply computer system, and was easily transferred to the personal computer prototype software. The program also allowed the user to enter new data or edit the current data in the records. Data was presented in groupings of like data in simple terms, with a minimum of

supply computer codes. Additionally, information from several sources was presented on the screen at the same time, to provide a clear understanding of relationship between the data. The program also printed optional reports for the user to provide a paper record of data.

Third, what is the appropriate computer language for system development that will be efficient, effective, and still maintain user-friendliness? The literature review examined the available database management system languages. Several languages were available, but dBASE (tm) was chosen because it was used previously to build other database management system applications. A variant of the dBASE language, called Clipper (tm) was selected on the basis of price, availability and expanded capabilities. It was readily available to the researcher, with a compiler and technical support.

Fourth, what form should a database take to efficiently organize WRM data? The databases were relational, because the structures are the easiest to understand and work with. The data was split into four separate databases to group like information together. Index files were added to increase the speed of searches, and decrease the time to recall data.

Fifth, will the program need to access and work with existing databases or computer systems? The prototype software needed data that was available in the supply computer system. A direct link with the SBSS computer would

have been very complicated and beyond the capability of the researcher. The transfer of data to the prototype software was accomplished by extracting a data file from the mainframe to a floppy disk, and then loading the file into the prototype software program. Other possible computer sources of data were not readily available.

Sixth, what are the criteria to ensure successful implementation and acceptance by base-level logisticians? For the software program to be successful, it had to provide an advantage over current techniques of managing WRM programs. The program needed to reduce the time spent leafing through paper reports, and reduce the time spent tracking, inspecting, and managing WRM program data. The evaluation of the program by base logisticians needed at least seventy-five percent acceptance for program success.

Summary of Chapter IV

The requirements collected from the user interviews were outlined, along with the program functions developed from those interviews. The requirements were to handle WRM authorizations, supply document data, inventory balances, and inspection information. Also, reports were needed to list the available data. In some cases, data from different areas was combined into a single report. The program required an interface with the standardized base supply system (SBSS) to bring in data, and a means to backup and recover program data. The first version of the program was evaluated by an experienced WRM manager to identify flaws in

the program. The evaluators recommended the program's objective be limited to consumable items. They recommended some changes to the program functions, screen appearance, and the reports and listings. Changes to the program were included, and the program was then sent to a larger group of users for evaluation.

V. Conclusion and Recommendations

Introduction

This chapter covers the response of the users to the evaluation of the program and User's Guide. Of the fifteen evaluation packets returned, thirteen of the users decided the program would be useful in managing a WRM program. Also recommendations for improvements in program functions, help screens, and the User's Guide are presented. The users recommended the program should be further developed, incorporating an interface file with the SBSS, and additional programming techniques to enhance program appearance.

Results

Of the seventeen evaluation packets sent, fifteen responses were recorded. Of the responses, thirteen users decided the program would be helpful in managing a WRM program. This represents an 86.7% acceptance rate. Based on the number of positive returns, the research project is successful.

Several of the users expressed recommendations for improvements in program functions. They also reported problems with the software that are unique to the hardware and software setups on their computers. The most needed program improvement is more context sensitive help screens for each of the options and an expanded User's Guide. Also,

they suggested an expansion of the program scope to include equipment and other non-consumable WRM items. Some software programs cause problems when run in conjunction with this program. A solution to this problem must be found before the program can be released for general use.

Help screens are available for each program function, and were well received by the users. Within each function, there are also help screens for each menu option. The problem is that the help screen for each menu option is not activated until an option is selected. Within each option, help screens are also available, but like the main menu, the help screen is not available for each of the menu options until it is selected. This is a result of the Clipper (tm) programming language, and cannot easily be corrected.

The User's Guide needs to be expanded. The guide assumes too much computer knowledge by the users. Although there is an automated installation for the program, there should also be an explanation of how to install the program onto a different hard disk drive and add the appropriate information to the system setup files. Also, the User's Guide should have more examples of screens from the program to familiarize the user with what to expect from the program. Several critical screens were overlooked.

The user needs a means to select from the available national stock numbers for use in the search function. This need is also applicable to the supply document function, where the search is performed by document numbers. This

recall can be done through a pop-up box to list available NSN or document numbers. The user can then select a NSN or document number by simply scrolling through the list, highlighting the desired item, and pressing the ENTER key.

At least one user reported problems with valid keyboard entries that were ignored and other non-valid keyboard commands that caused the program to terminate unexpectedly (14). The problems occurred because of conflicts with the keyboard cursor keys and the numeric keypad (which can also act as cursor keys). Although the program was tested on a Zenith Z-248 computer, and an 80386 computer system, the use of some memory-resident programs and the system setup can interfere with the program operation. Any further program development should try to better handle the different operating environments or advise the user of the target system for the program.

Conclusions

The program developed from this research shows that a WRM software program is a useful tool in managing a WRM program. While this initial program addresses a specific need in the logistic plans community, there is room for improvement. The program developed from this research is too limited in scope and capability, and requires further requirements research and expanded programming efforts.

Recommendations

This program needs further development before it can be used on a daily basis. Expansion of the program functions and better documentation are required.

Consideration should be given to incorporating the results of this research into an existing program, such as HEIMS (Housekeeping Equipment Inventory Management System).

An interface file should be established with the concurrence of the AFLMC and SSC to maintain compatibility of the data file between the SBSS and SIMBL. In this way, changes to the SIMBL program would be reflected in the data file extracted from the SBSS computer.

The development of a software program involves hundreds of man-hours in writing the code and debugging. There were many programming techniques that were not used in order to get the program completed within the time limit of the research project. One technique not used was 'pop-up' menus, which create a box filled with data and allows the user to highlight an item and select that item. These techniques could have made the program easier to use, as well as providing the 'bells and whistles' that make a program more enjoyable.

After further program development, the software should be tested at several bases for an extended trial period. A test period of several weeks may identify deficiencies or flaws that could not be observed with the sample database.

Summary of Chapter V

This study validated the need for development of a computer application supporting the War Reserve Materiels program. The study investigated the requirements for development of a software program for managing WRM consumable asset data. From the requirements, a prototype database management system application was developed, tested, and sent to a group of users for evaluation.

Seventeen evaluation packets were sent to users in five major commands, with fifteen responses recorded. Of the responses, thirteen of the users decided the program would be helpful in managing a WRM program. Based on the response rate of 86.7%, the program is a success. This research is important because it is an example of enhancing base LGX operations through computer management of available data. In addition, the methodology used proved useful for developing the prototype software. The benefit of developing this application is a potential savings of management time energy through better asset inventory accuaracy. Similar software applications could be developed using the methodology.

Several of the users expressed recommendations for improvements in program functions, help screens, and the User's Guide. The program should be further developed, incorporating an interface file with the SBSS, and additional programming techniques to enhance program appearance.

Appendix A: Dictionary of Terms

AFLMC	Air Force Logistics Management Center at Gunter AFB; responsible for development of applications for large computer systems.
AFR	Air Force Regulation
Authorization	A declaration of an item as a WRM item, but not necessarily funded
BLSS	Base Level Self-Sufficiency Spares
Clipper	An expanded version of the dBASE programming language resulting in a compiled program.
Compiler	A software program that converts a text program to executable machine code.
Condor	A programming language similar to dBASE, it is an interpreted language.
Consumable	Expendable items directly related and necessary for a weapon system or combat support.
Data	Any type of information, usually classified as character, numeric, logical, or date.
Database	A collection of records about a related subject; the records are usually physical or electronic.
dBASE	A database management system and programming environment. It forms the basis for several other DMBS programming languages. dBASE is an interpreted language.
DBMS	Database Management System
DIFM	Due In From Maintenance
Equipment	Items which do not lose their individual identity when used.
Expendable	Items not designed to be repaired or reused.
FORTRAN	A computer programming language developed in the late 1970's used primarily for scientific applications with large computational needs.

FoxBASE	An expanded version of the dBASE programming language resulting in a compiled program.
GSA	General Services Administration
Harvest Bare	A series of packages (Harvest Eagle, Harvest Falcon) designed to support aircraft operations at locations with little prior preparation.
Hierarchial	A database structure where each record has one parent record, but has many child records. Often described as a tree-like 'root' structure.
HEIMS	House-keeping Equipment Inventory Management System
Interpreter	A software program that evaluates a single line of text program, executes the command, then evaluates the next line. It is a slower method of running a program, but allows for easier debugging.
LCD	Liquid Crystal Display
LGX	Office symbol for Logistics Plans and Programs office
Logistics	The management science of acquiring, storing, transporting, and maintenance of supplies, equipment, and personnel.
MAC	The Military Airlift Command
MAJCOM	Major Command
Materiel	Same as material
Munitions	A general class of items that includes bullets, bombs, missiles, flares, chaff, pyrotechnics, fuzes, and other explosive devices.
Network	A database structure where each record can have many parent records and many child records. This complex structure is often described as a 'neural net'.
NSN	National Stock Number - a 15 alphanumeric code that uniquely designates any item used in the standardized base supply system.

OOP	Object Oriented Programming - a technique for defining portions of computer code as objects, and for handling the relationships between objects. This technique is used in many newer computer programs.
PACAF	The major command designator for the Pacific Air Forces
PACOPS	The major command office symbol for the Operations section of PACAF.
Pascal	A computer programming language named for a french mathematician, the language is used primarily for applications that manipulate text, such as word processors.
POC	Point Of Contact
POMCUS	Pre-positioning of Materiel Configured Unit Sets; an Army program similar the Air Force WRM program.
PPP	Pre-Positioned Packages of assets stored at a location in anticipation of a need by a deploying force
QuickSilver	An expanded version of the dBASE programming language resulting in a compiled program.
QLP	Query Language Program - one of the programming languages used on the Sperry 1160 computer system as part of the SBSS.
Relational	A database structure where each record can have only one parent record, and one child record. Often described as a table of data.
Reparable	Items designed to be repaired or reused.
R07	The numeric designation for the WCDO/BLSS spares report.
SAC	The major command designator for the Strategic Air Command
SBSS	Standardized Base Supply System
SIMBL	System for Inventory Management for the Base Logistician
SSC	Standard Systems Center at Gunter AFB; responsible for development of applications for small computer systems.

Stockpile	A technique of collecting and storing an item in anticipation of a future increase in demand for the item.
TAC	The major command designator for the Tactical Air Command
tm	trademark
USAF	The Department of Defense designator for the United States Air Forces
USAFE	The major command designator for the United States Air Forces in Europe
WCDO	Wartime Consumables Distribution Objective - establishes the basic WRM requirements for each unit.
WPARR	War Plans Additive Requirements Report - additional items designated as WRM by the major command.
WRM	War Reserve Materiel
WRSK	War Readiness Spares Kits - transportable packages of equipment and aircraft spares used to support operations on a daily basis or at a deployed location.
Z248	An IBM/AT (80286) compatible computer made by Zenith Data Systems, it is found in many logistics plans and program (LGX) offices.

Appendix B: User's Guide

Overview of Program

This program was developed as a tool for the base logistician who manages a large WRM program. This program is designed to handle only consumable items, those with an ERRC code of XB3. It allows the user to build databases of authorizations and supply information without maintaining files of paper printouts. It coordinates authorizations, supply details, and organizational data and presents it in a meaningful way. This program will record inspection dates, discrepancies, references and user responses so they can be recalled later. Also, a variety of useful reports can be generated that consolidate existing data.

Required Hardware

- An IBM PC/XT/AT or compatible computer
- 512K RAM or greater
- A Hard disk with at least 1Mb available space on Drive C:
- A floppy drive designated as Drive A:

Recommended Hardware

- A color monitor
- A printer attached to LPT1:

Installation

1) At the DOS prompt, put the program diskette in the floppy drive, close the door and type the command: A:INSTALL.

2) The program will automatically create a C:\SIMBL directory and copy the program files from the floppy.

3) Add the SIMBL directory to the path specification of your Autoexec.bat file.

Program Functions

Main Menu. The program will begin with a display of the main menu. You use this menu to select which area of the WRM information you want to work with. The highlighted box is the currently selected program function. You move from box to box by pressing any of the arrow keys or the space bar. Pressing the Up arrow key, the Right arrow key, or the space bar will advance the highlight to the next box. Pressing the Down arrow or Left arrow keys will move the highlight back to the previous box. When you reach the bottom right or top left box, the highlighted box will move to the opposite corner. Try it! Pressing the Enter key will take you to one of the nine functions described in this section.

Authorizations. The authorizations screen displays information about a WRM authorization, such as the source document number you have established, the authorized quantity and the actual quantity on-hand. The authorized

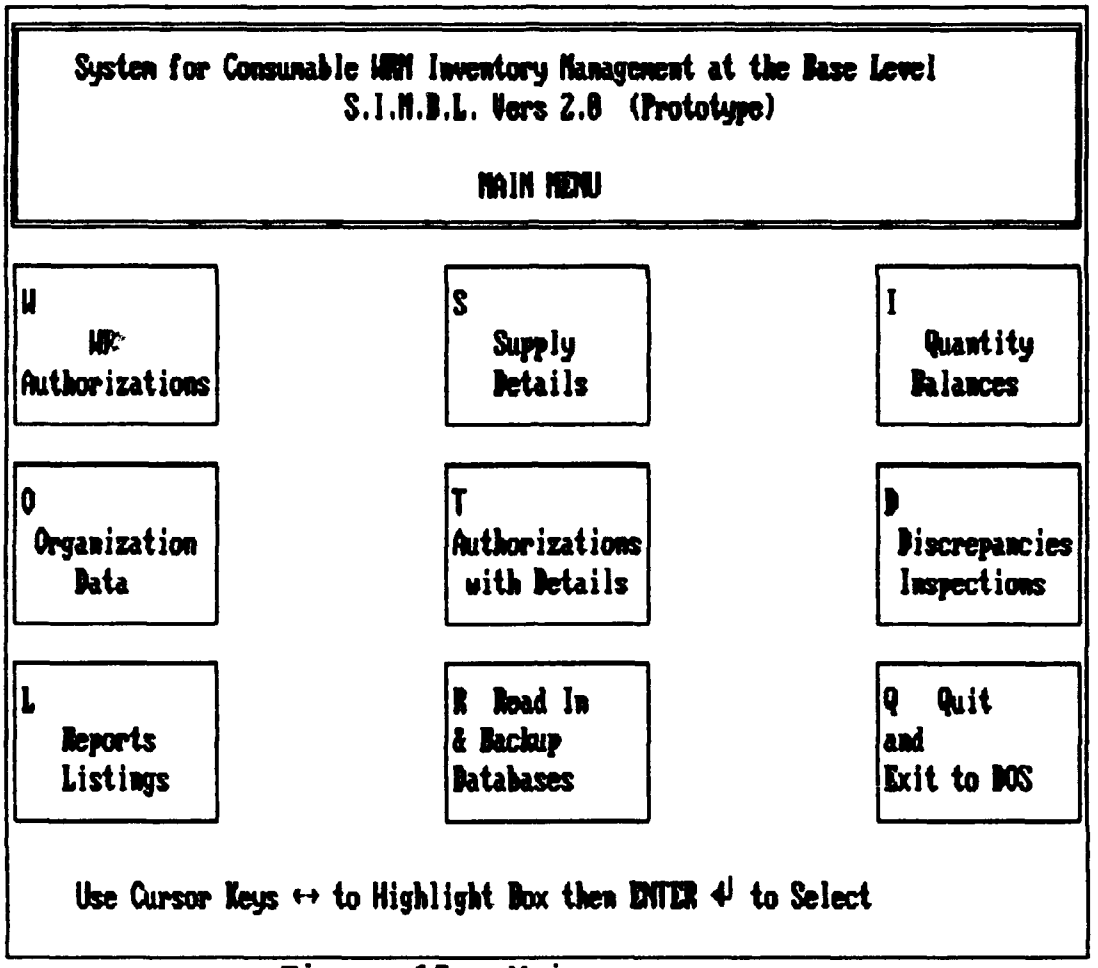


Figure 15. Main menu screen

quantity is the net quantity authorized (authorized - unsupportable). The screen also displays the unit value of the item and the total value of all units. The sample screen shows 96 quarts of hydraulic fluid, which are valued at $11.71 each, for a total of $1124.16. The ERRC Code should be XB3 for all expendable, consumable items. It is provided for program expansion at a later date.

The bottom line lists the different options available for that screen. Next will call up the next record in the database, while Prev will move 'backwards' to the previous

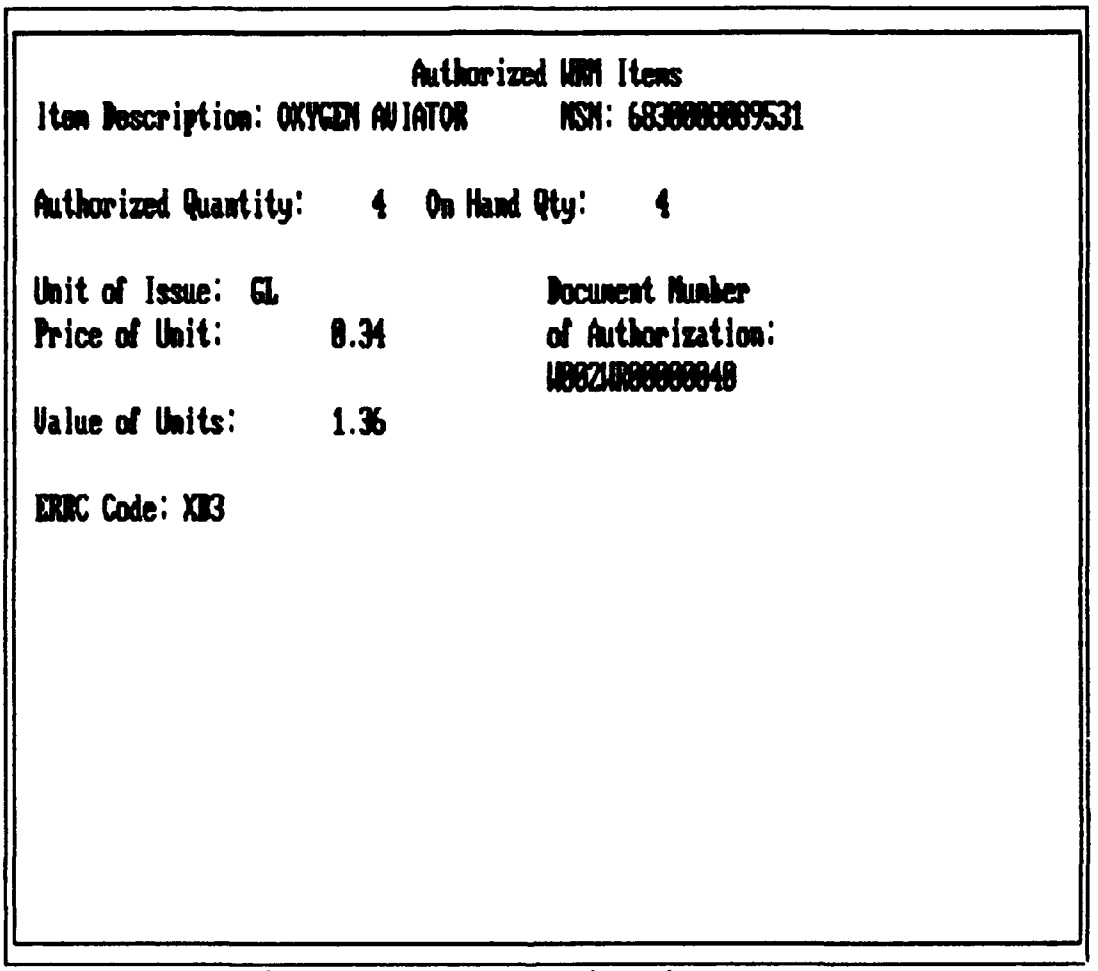


Figure 16. Authorization screen

record. First will go to the top of the database, and Last will move to the bottom of the database. A record can be located by National Stock Number (NSN) by selecting the Search option. Edit will allow you to make changes to the data on screen, and Append will let you add a new item and authorization to the database. Delete is used to mark a record for deletion. It does not actually remove the data until you exit the program, and will prompt you to make sure you really want to delete that record or other records.

Quit will exit the screen function and take you back to the Main Menu.

Help is available in the Search, Edit and Append functions by pressing the F1 key.

Supply Documents. The supply details screen is used to maintain information about supply documents.

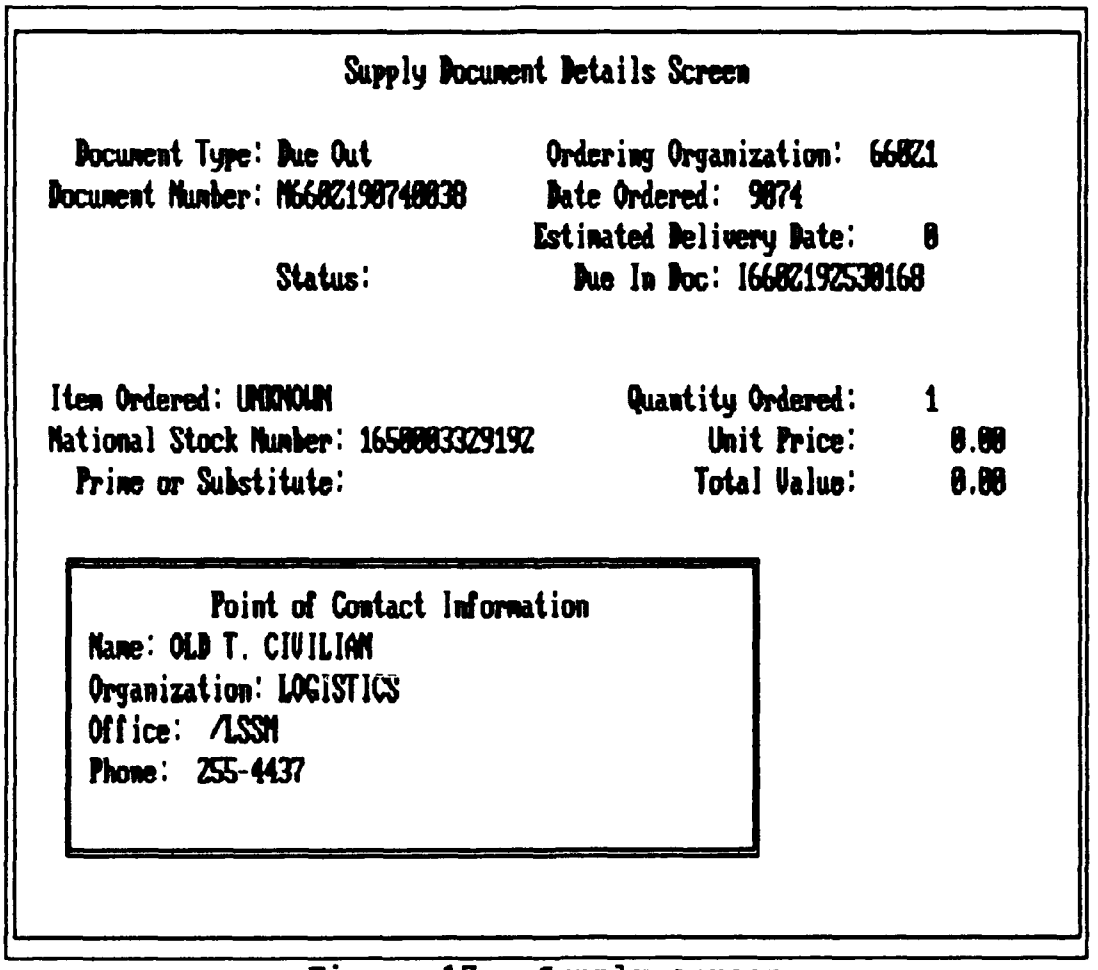


Figure 17. Supply screen

These documents should be WRM items on order to fill a shortage. This screen breaks the Due-Out document number into several pieces and uses them to look up other related

information. For example the first six characters tell the program what kind of document it is, and what organization submitted the document to Base Supply. If the program can find the Organization's Org/Shop code, it displays any point of contact it has. The dates given are Julian dates (ie, 0102 is the one hundred and second day of the year 1990). The program does not translate the Julian date to normal Gregorian dates (YY/MM/DD).

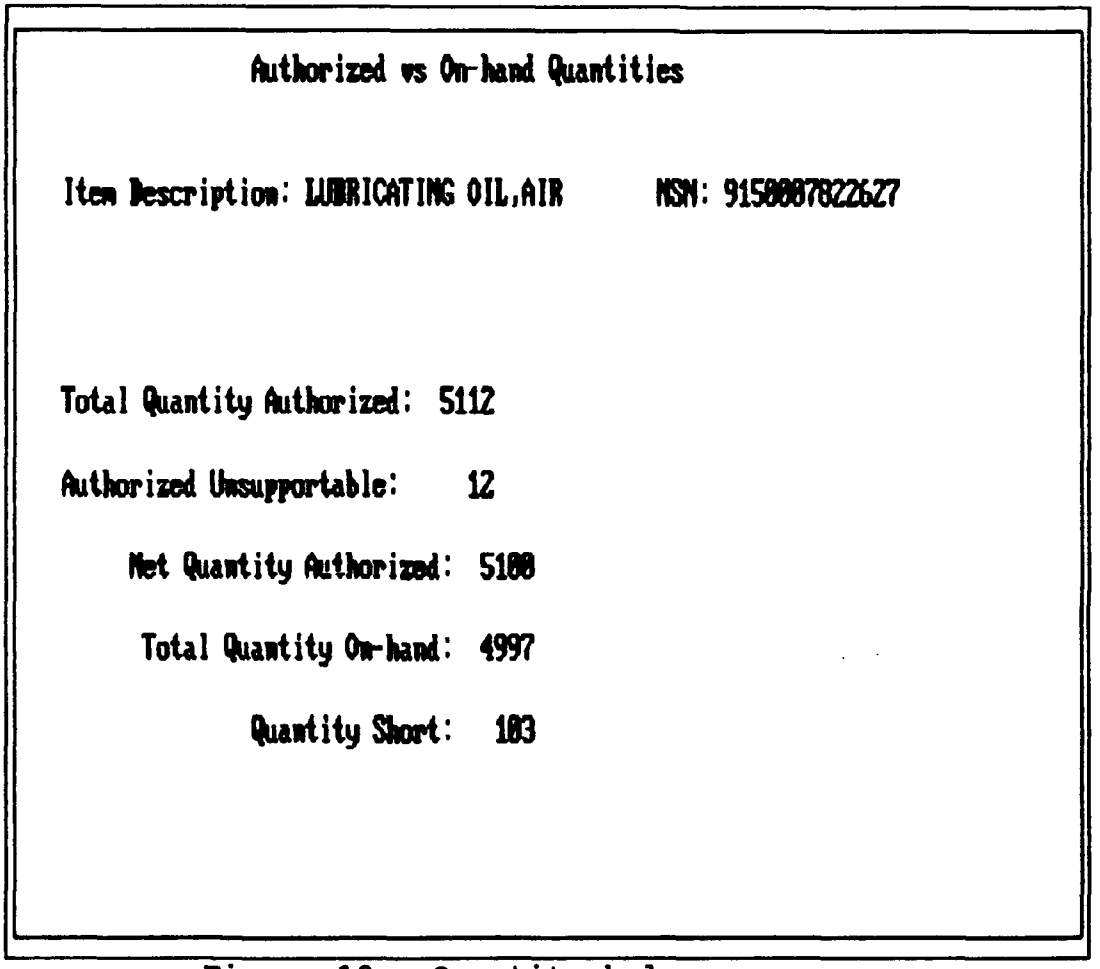


Figure 18. Quantity balances screen

Quantity Balances. The quantity balances screen is used to specify information about the quantity authorized,

authorized unsupportable amounts, the quantity on-hand, and the quantity short for each item. When you add a new item in the Authorization function, the program assumes the quantity authorized is the bottom line, that is there is no authorized unsupportable quantity. If you need to specify that value, this option will allow you to do so using the Edit or Append options. Also, the program will maintain the on-hand quantity and from the available data, calculate the shortage quantity.

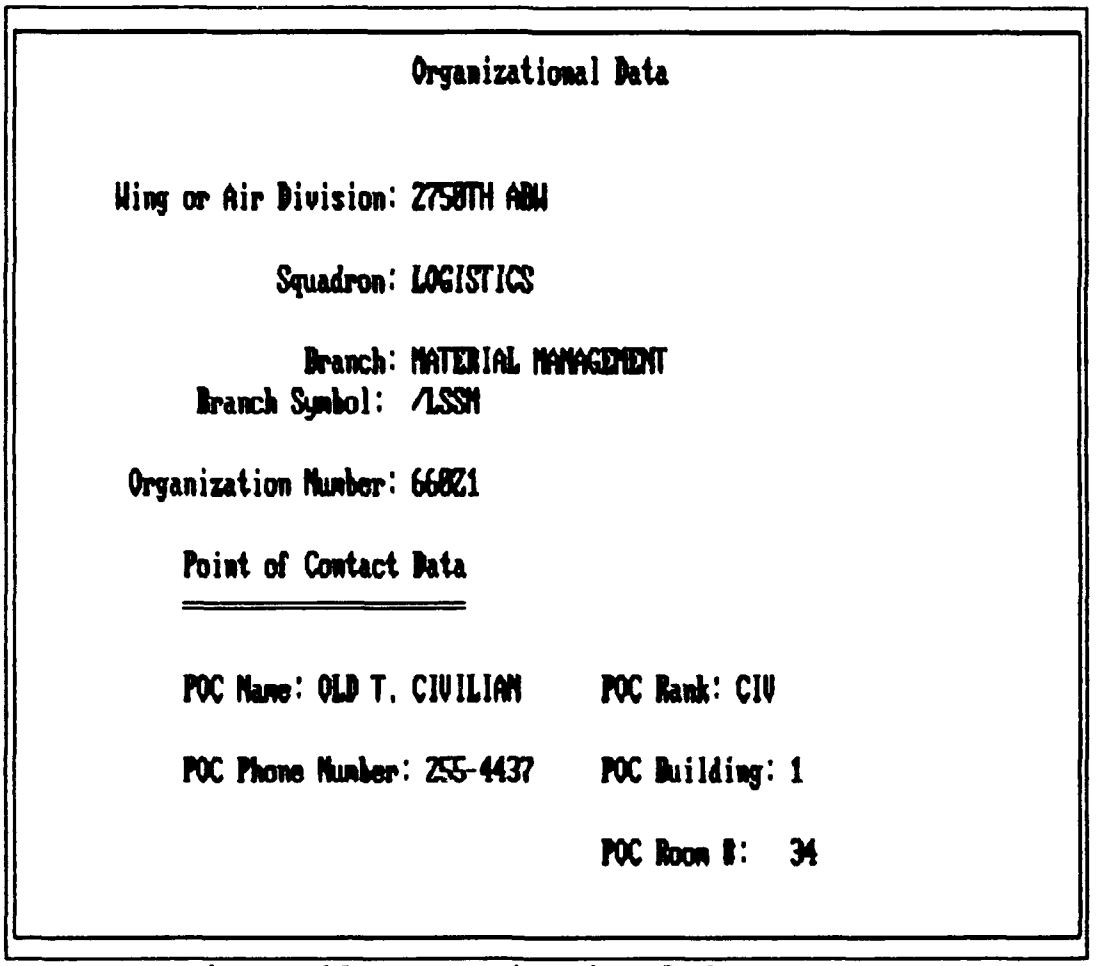


Figure 19. Organizational data screen

Organizational Data. This function builds a file of org/shop codes along with point of contact information. Some of the information is printed in reports, while other data is displayed with supply document (details) information. This saves the user from having to look up the Org/Shop code or stock number for the item just to find out who is holding the material. This data is kept when you read in new SBSS data, and does not go away unless you specifically delete the record.

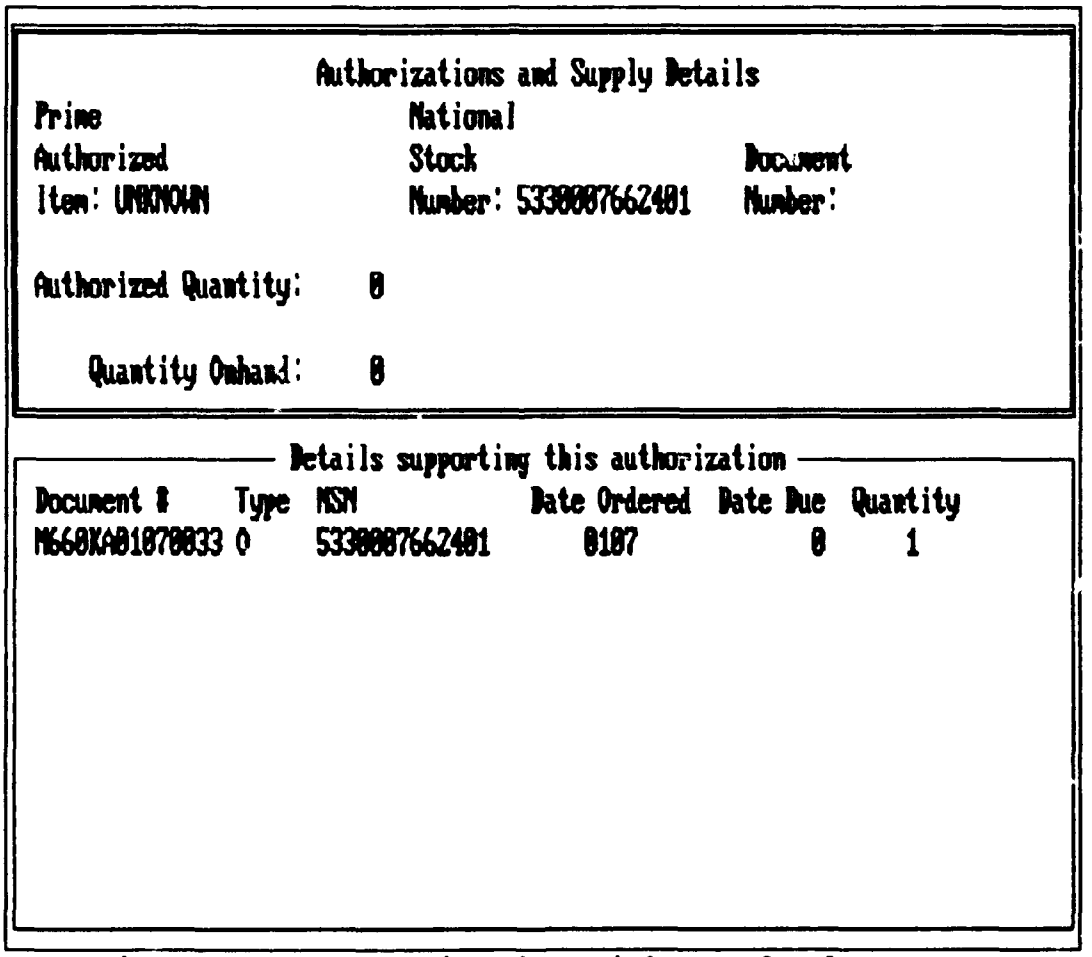


Figure 20. Authorization with supply documents

Authorizations with Details. Coordinating WRM authorizations with Supply documents is the purpose of this function. Authorizations and Stock Numbers appear in the top half of the screen. The View option is used to look for Supply documents (details) that correspond to the NSN in the top half of the screen. If there are any Supply Due-Out documents that have the same National Stock Number, they will appear in a second window in the lower half of the screen. The Supply document number, Julian date, due date and quantity are given on the same line.

Inspection and Discrepancies. This function is probably the most useful of all the program options. You can make notes of current and past inspection results, keep track of inspection dates, and responses from the inspected organization. Each Discrepancy, Reference, and Response can be as long as 64,000 characters, so you can make long memos.

During an edit or append, the user can enter a long note in the Discrepancy, Reference, and Response fields. A sample window of data in the Discrepancy entry is shown in the figure. All three types of notes: Discrepancy, Reference or Response can be as long as 64,000 characters. If the User types to the end of a line, the text will automatically wrap around to another line.

If you enter a 'Y' or 'y' to the Follow Up field, the program will automatically calculate a reinspection date for 30 days from the current date. The dates are in the format of MM/DD/YY, and will appear on the inspections report.

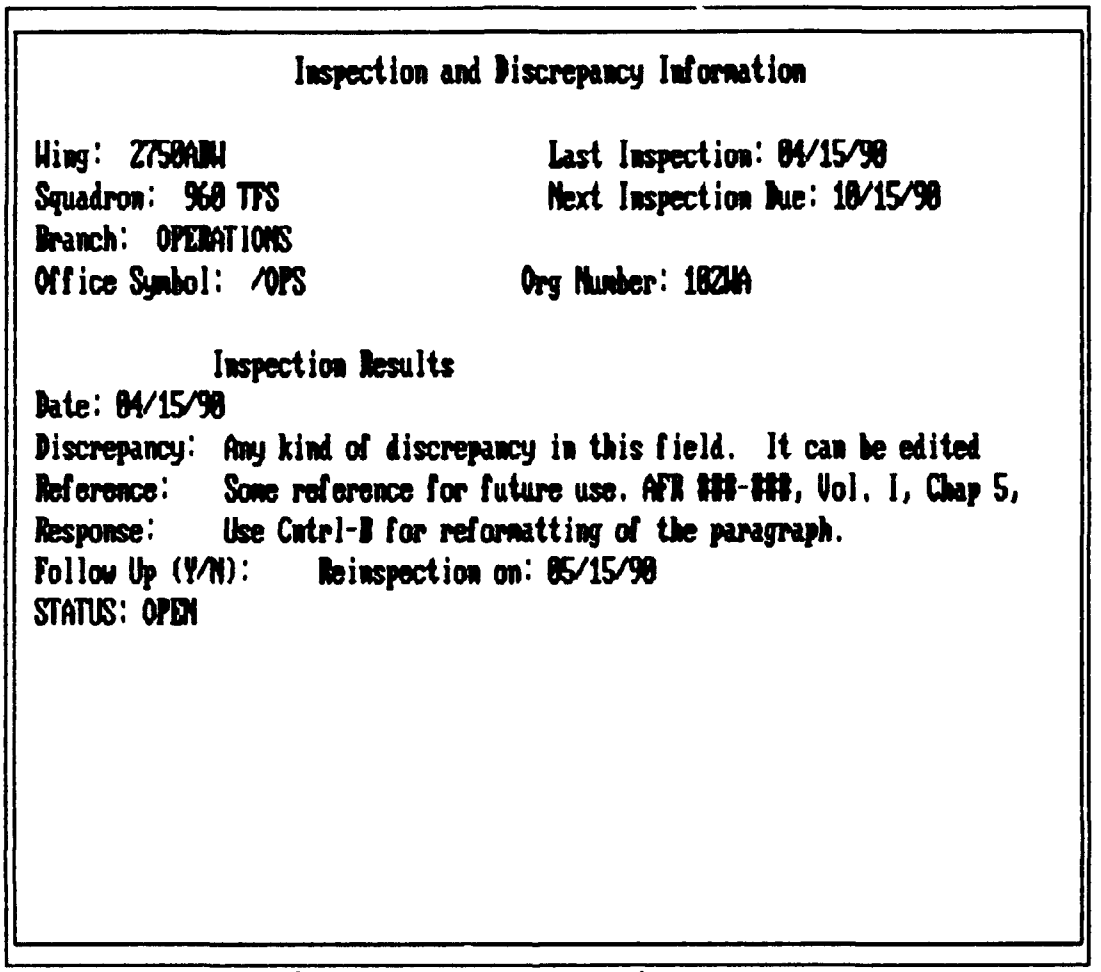


Figure 21. Inspection screen

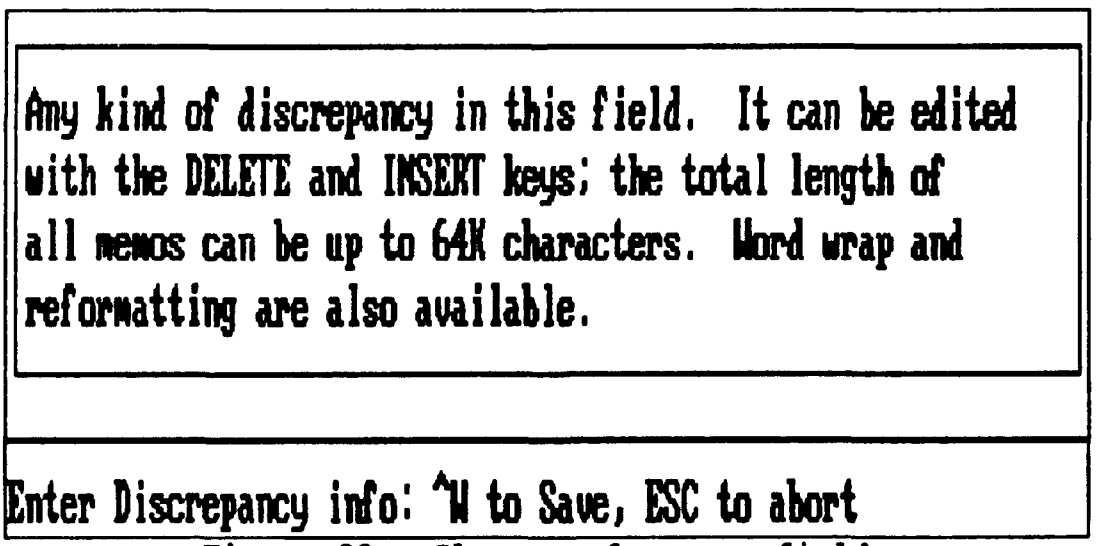


Figure 22. Closeup of a memo field

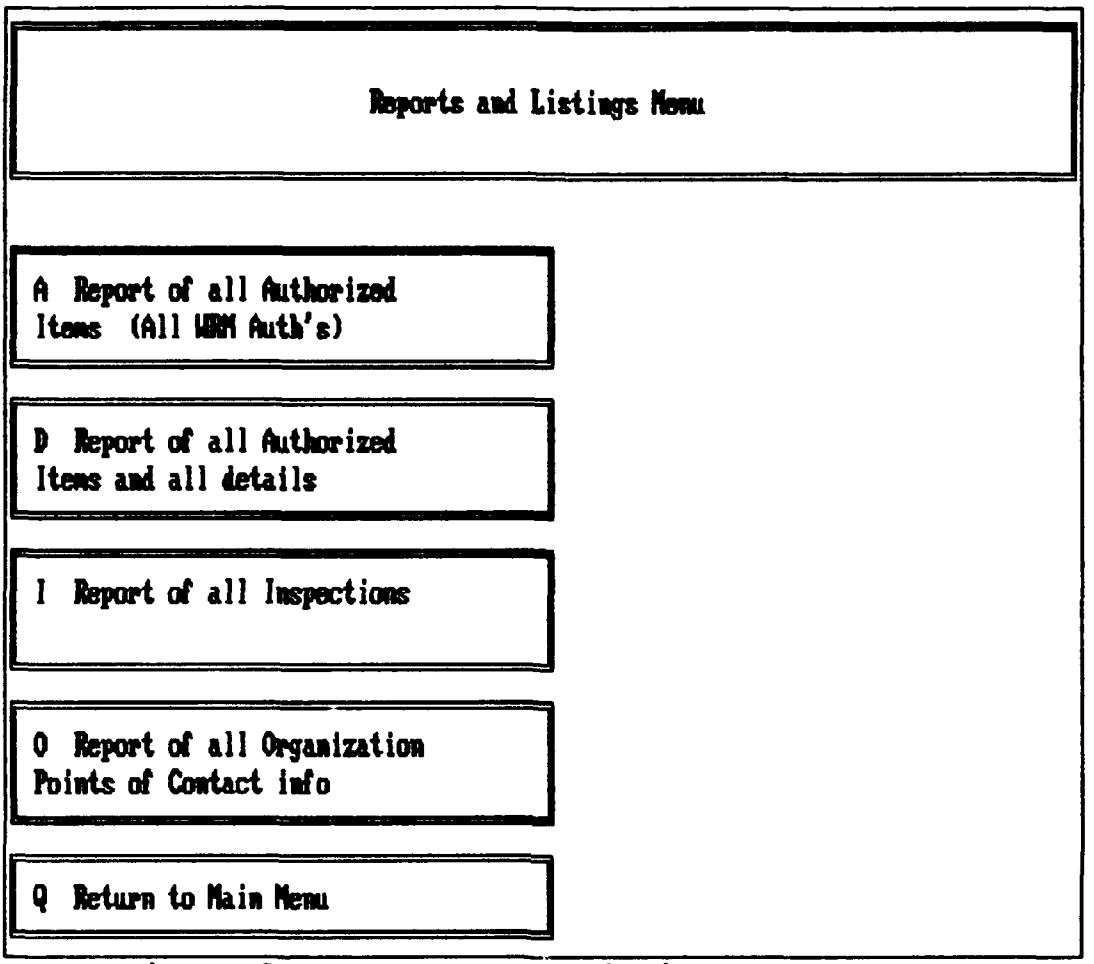


Figure 23. Reports and listings menu screen

Reports and Listings. You will be presented a menu of reports that can be generated and sent to the screen and the printer. The User selects from this menu in the same manner as the Main Menu. The first report generates a report of all the available information from the Authorizations screen. The second report is useful for creating a report that lists each WRM item and all the Supply Document Details available. The third report lists all the organization inspection data, while the fourth reports all the Point of Contact information for each organization.

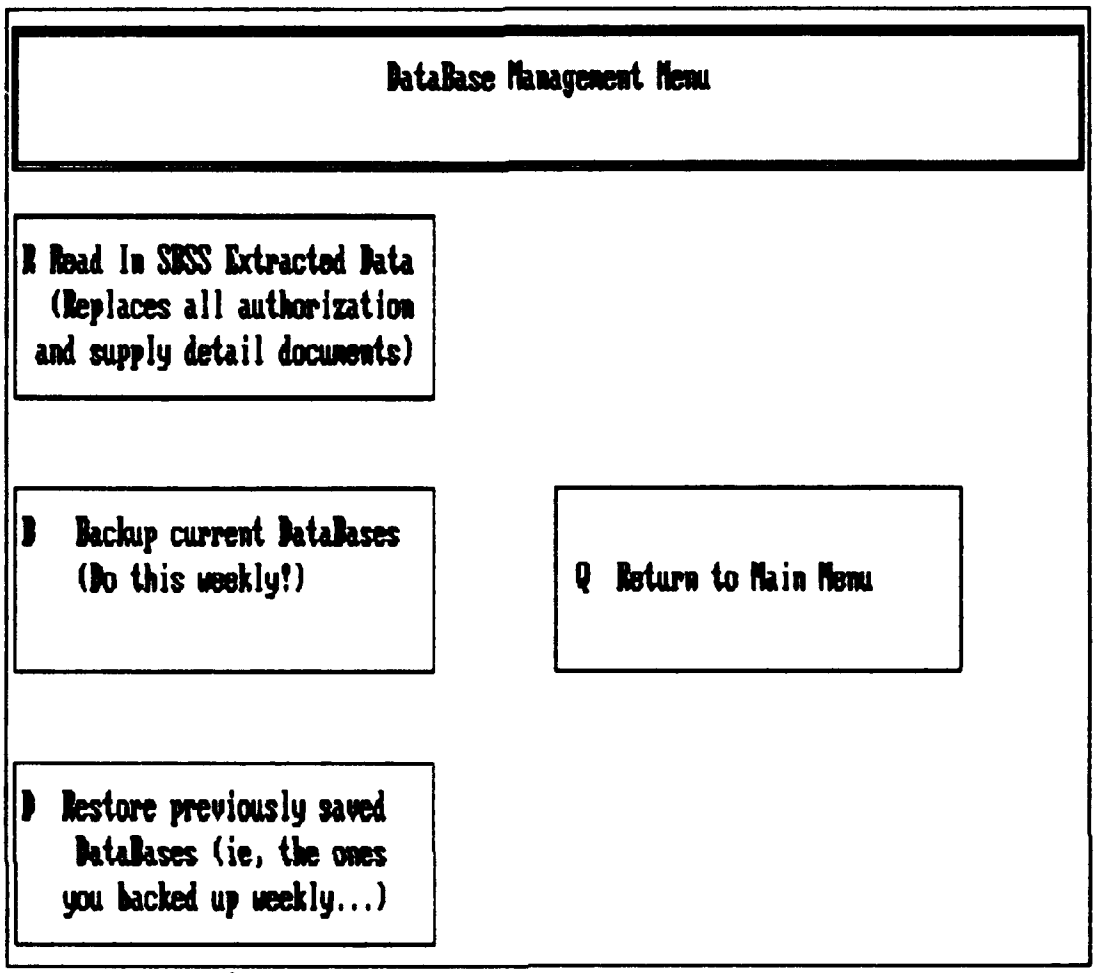


Figure 24. Database management menu

Read In and Backup Databases. With this function, the User can bring in data from the Supply Base Computer and thereby create new databases of authorization and supply document information. See the section entitled 'Extracting SBSS Data' for how to get data from Supply onto a floppy disk to use with this program. The User can also backup current data to a floppy disk, or restore data from a previously backed up floppy disk.

Quit Program. Will exit the program, deleting marked records and cleaning up any loose ends. The program will

give you a last chance to recover records that you have marked for deletion. If you confirm the deletion by pressing the 'Y' key, the program will remove that information from the databases. If you do want to delete, and you confirm the deletion you will erase only the records marked for deletion, this data cannot be recovered! If you do not want to delete information, but confirm the deletion, the data will be gone and the only way you have to get it back is to re-enter it the hard way.

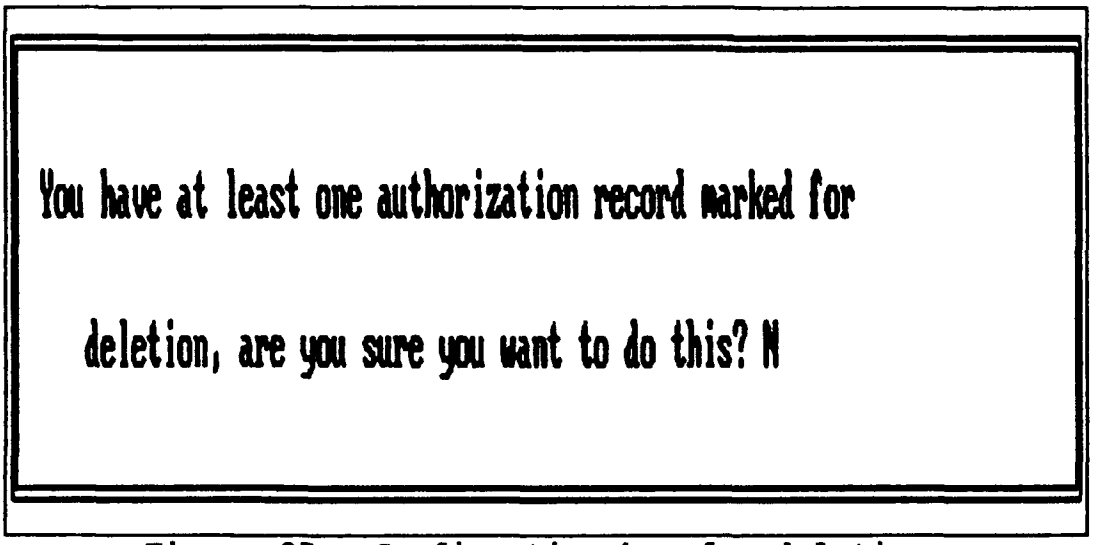


Figure 25. Confirmation box for deletions

The bottom line here is that if you are unsure about deleting any information, press 'N' (or any key except 'Y'), and then all the records marked for deletion will be recovered.

Using the Help function

Pressing the <F1> key will access the Help function built into the program. The help information consists of a

rectangular window that pops up on the screen, and displays some additional information about the current function. It does not work during the opening screen, the Report & Listings screen, or the Readin Databases screen.

Extracting SBSS Data

NOTE: To save you time and energy, the developer has included sample databases so you can experiment on the different program functions with real data.

There is a separate program on the diskette, which should also be in the SIMBL directory, called WRM.PRG. This file is a QLP program written to pull data from the SBSS 1100/60 computer. Make a copy of the WRM.PRG file and take it to the computer operator at Base Supply. The operator can upload the program, run it, and download the resulting file. The output file can be loaded into the SIMBL program using the ReadIn function. When you access the ReadIn function, put in the diskette, and the program will do the rest!

Catastrophic Disaster

Into every life a little rain must fall, and at some point you will probably have a catastrophic failure of either software or hardware on your computer. Your best defense against software failure is to backup your databases whenever you make changes to them, particularly after adding new information. Old data you can delete easily, the new data might have to typed in all over again! If disaster

should occur with the software, such as a file being damaged, copy your data backup disk into the C:\SIMBL directory. If the program still will not run, reinstall the program from the original disks, and then recover your current databases from the backup data disk. If none of these steps solve the problem, then it is likely there is a problem with the hardware of your machine.

This program is a prototype, it is not a full fledged program that will do anything and everything! From the comments of you and other logisticians, this may form the basis for an expanded version with greater capabilities and functions. Your comments are important to the success of this project, so please return the completed response card as soon as possible.

Captain Kevin M. Tanzer
AFIT/LSG
Wright-Patterson AFB, OH 45433
Home Phone: (513) 297-0551
Work Phone: AV 785-4437

Appendix C: Program Source Code

```
***
*** C:\CLIP\SIMBL.PRG : Main menu.
*** Generated March 18, 1990 from C:\UI\WW\MAINMENU.WW
*** Target environment: Clipper Summer 87
***

* environment
SET SCOREBOARD off
SET CONFIRM off
SET CURSOR OFF
SET KEY 28 TO Help

* Database flags for PACK/REINDEX
PUBLIC
mBASIC,mINSPECT,mORGDATA,mDOCUMENT,mBACKUP,mLastOrg,help_cod
e;
,mLastArea
mBASIC = .F.
mINSPECT = .F.
mORGDATA = .F.
mDOCUMENT = .F.
mBACKUP = .F.
help_code = "simbl"
c_normal = "W+/N,B+/N,N,N,B+/N"
c_selected = "W+/N,W+/B,N,N,B+/N"
c_unselected = "B+/N,B+/W,N,N,B+/N"
c_flashing = "R*/W,B+/N,N,N,B+/N"

* menu initialization
PRIVATE optkeys, numopts, oldchoice, newchoice, key
optkeys = "WSIOTDLRQ"
numopts = 9
oldchoice = 0
newchoice = 1
key =0

DO Setup
* initialize databases

* inkey aliases
PRIVATE RKup,RKdn,RKrt,RKlt,RKret,RKspc
RKup = 5
RKdn = 24
RKrt = 4
RKlt = 19
RKret = 13
RKspc = 32

* Reminder about not using classified data
DO ClassScr
```

```
* main loop: iterates once for each time an option action is
performed
DO WHILE .t.

  * display the menu screen
  SET COLOR TO &c_normal
  CLEAR
  @ 22, 5 SAY "Use Cursor Keys "+chr(27)+chr(26)+" to
Highlight Box then ENTER "+chr(17)+"┘ to Select"
  SET COLOR TO &c_normal
  @ 0, 0, 5, 79 BOX "┌┐│┘─└│"
  SET COLOR TO &c_selected
  @ 1,1,4,78 BOX "         "
  @ 1, 5 SAY "System for Consumable WRM Inventory Management
at the Base Level"
  @ 2, 25 SAY "S.I.M.B.L. Vers 2.0  (Prototype)"
  @ 4, 35 SAY "MAIN MENU"
  SET COLOR TO &c_selected
  @ 6, 0, 10, 15 BOX "┌┐│┘─└│ "
  SET COLOR TO &c_unselected
  @ 7, 1 SAY "W"
  @ 8, 6 SAY "WRM"
  @ 9, 1 SAY "Authorizations"
  SET COLOR TO &c_selected
  @ 6, 32, 10, 47 BOX "┌┐│┘─└│ "
  SET COLOR TO &c_unselected
  @ 7, 33 SAY "S"
  @ 8, 36 SAY "Supply"
  @ 9, 36 SAY "Details"
  SET COLOR TO &c_selected
  @ 6, 64, 10, 79 BOX "┌┐│┘─└│ "
  SET COLOR TO &c_unselected
  @ 7, 65 SAY "I"
  @ 8, 68 SAY "Quantity"
  @ 9, 67 SAY "Balances"
  SET COLOR TO &c_selected
  @ 11, 0, 15, 15 BOX "┌┐│┘─└│ "
  SET COLOR TO &c_unselected
  @ 12, 1 SAY "O"
  @ 13, 2 SAY "Organization"
  @ 14, 5 SAY "Data"
  SET COLOR TO &c_selected
  @ 11, 32, 15, 47 BOX "┌┐│┘─└│ "
  SET COLOR TO &c_unselected
  @ 12, 33 SAY "T"
  @ 13, 33 SAY "Authorizations"
  @ 14, 34 SAY "with Details"
  SET COLOR TO &c_selected
  @ 11, 64, 15, 79 BOX "┌┐│┘─└│ "
  SET COLOR TO &c_unselected
  @ 12, 65 SAY "D"
  @ 13, 66 SAY "Discrepancies"
  @ 14, 66 SAY "Inspections"
```

```
  SET COLOR TO &c_selected
  @ 16, 0, 20, 15 BOX "┌┐│┘─└│ "
  SET COLOR TO &c_unselected
  @ 17, 1 SAY "L"
  @ 18, 4 SAY "Reports"
  @ 19, 4 SAY "Listings"
  SET COLOR TO &c_selected
  @ 16, 32, 20, 47 BOX "┌┐│┘─└│ "
  SET COLOR TO &c_unselected
  @ 17, 33 SAY "R  Read In"
  @ 18, 33 SAY "& Backup"
  @ 19, 33 SAY "Databases"
  SET COLOR TO &c_selected
  @ 16, 64, 20, 79 BOX "┌┐│┘─└│ "
  SET COLOR TO &c_unselected
  @ 17, 65 SAY "Q   Quit"
  @ 18, 65 SAY "and"
  @ 19, 65 SAY "Exit to DOS"

  * keyhit loop: iterates ones for each key input, breaks on
selection
  DO WHILE .t.

    * if selected option has changed, update the bounce-bar
    IF oldchoice<>newchoice
      * highlight new option
      DO CASE

     CASE newchoice =1
          SET COLOR TO &c_normal
          @ 6, 0, 10, 15 BOX "┌┐│┘─└│"
          SET COLOR TO &c_selected
          @ 7,1,9,14 BOX "          "
          @ 7, 1 SAY "W"
          @ 8, 6 SAY "WRM"
          @ 9, 1 SAY "Authorizations"

     CASE newchoice =2
          SET COLOR TO &c_normal
          @ 6, 32, 10, 47 BOX "┌┐│┘─└│"
          SET COLOR TO &c_selected
          @ 7,33,9,46 BOX "          "
          @ 7, 33 SAY "S"
          @ 8, 36 SAY "Supply"
          @ 9, 36 SAY "Details"

     CASE newchoice =3
          SET COLOR TO &c_normal
          @ 6, 64, 10, 79 BOX "┌┐│┘─└│"
          SET COLOR TO &c_selected
          @ 7,65,9,78 BOX "          "
          @ 7, 65 SAY "I"
          @ 8, 68 SAY "Quantity"
          @ 9, 67 SAY "Balances"
```

```
CASE newchoice =4
     SET COLOR TO &c_normal
     @ 11, 0, 15, 15 BOX "┌┐ │┘─└│"
     SET COLOR TO &c_selected
     @ 12,1,14,14 BOX "          "
     @ 12, 1 SAY "O"
     @ 13, 2 SAY "Organization"
     @ 14, 5 SAY "Data"

CASE newchoice =5
     SET COLOR TO &c_normal
     @ 11, 32, 15, 47 BOX "┌┐ │┘─└│"
     SET COLOR TO &c_selected
     @ 12,33,14,46 BOX "          "
     @ 12, 33 SAY "T"
     @ 13, 33 SAY "Authorizations"
     @ 14, 34 SAY "with Details"

CASE newchoice =6
     SET COLOR TO &c_normal
     @ 11, 64, 15, 79 BOX "┌┐ │┘─└│"
     SET COLOR TO &c_selected
     @ 12,65,14,78 BOX "          "
     @ 12, 65 SAY "D"
     @ 13, 66 SAY "Discrepancies"
     @ 14, 66 SAY "Inspections"

CASE newchoice =7
     SET COLOR TO &c_normal
     @ 16, 0, 20, 15 BOX "┌┐ │┘─└│"
     SET COLOR TO &c_selected
     @ 17,1,19,14 BOX "          "
     @ 17, 1 SAY "L"
     @ 18, 4 SAY "Reports"
     @ 19, 4 SAY "Listings"

CASE newchoice =8
     SET COLOR TO &c_normal
     @ 16, 32, 20, 47 BOX "┌┐ │┘─└│"
     SET COLOR TO &c_selected
     @ 17,33,19,46 BOX "          "
     @ 17, 33 SAY "R  Read In"
     @ 18, 33 SAY "& Backup"
     @ 19, 33 SAY "Databases"

CASE newchoice =9
     SET COLOR TO &c_normal
     @ 16, 64, 20, 79 BOX "┌┐ │┘─└│"
     SET COLOR TO &c_selected
     @ 17,65,19,78 BOX "          "
     @ 17, 65 SAY "Q   Quit"
     @ 18, 65 SAY "and "
     @ 19, 65 SAY "Exit to DOS"
```

```
      ENDCASE

      * reset oldchoice for another pass
      oldchoice =newchoice
    ENDIF

    * if return or an option trigger has been hit, perform
the option
    IF key=RKret .or. at(upper(chr(key)), optkeys) >0
      SET COLOR TO &c_normal
      * fall out to action loop
      EXIT
    ENDIF

    * get key input
    key=inkey(0)

    * update choice number based on key response
    DO CASE
      * down/right arrow: increment choice or wrap
      CASE key =RKdn .or. key =RKrt .or. key =RKspc
     newchoice = iif(oldchoice=numopts,1,oldchoice+1)
      * up/left arrow: decrement choice or wrap
      CASE key = RKup .or. key = RKlt
     newchoice = iif(oldchoice=1,numopts,oldchoice-1)
      * option trigger: set choice to option
      CASE at(upper(chr(key)), optkeys) > 0
     newchoice = at(upper(chr(key)), optkeys)
    ENDCASE

    IF oldchoice<>newchoice
      * lowlight old option
      DO CASE

     CASE oldchoice =1
          SET COLOR TO &c_normal
          @ 6, 0, 10, 15 BOX "┌┐│┘─└│"
          SET COLOR TO &c_unselected
          @ 7,1,9,14 BOX "          "
          @ 7, 1 SAY "W"
          @ 8, 6 SAY "WRM"
          @ 9, 1 SAY "Authorizations"

     CASE oldchoice =2
          SET COLOR TO &c_normal
          @ 6, 32, 10, 47 BOX "┌┐│┘─└│"
          SET COLOR TO &c_unselected
          @ 7,33,9,46 BOX "          "
          @ 7, 33 SAY "S"
          @ 8, 36 SAY "Supply"
          @ 9, 36 SAY "Details"

     CASE oldchoice =3
```

```
      SET COLOR TO &c_normal
      @ 6, 64, 10, 79 BOX "┌┐│┘─└│"
      SET COLOR TO &c_unselected
      @ 7,65,9,78 BOX "         "
      @ 7, 65 SAY "I"
      @ 8, 68 SAY "Quantity"
      @ 9, 67 SAY "Balances"

CASE oldchoice =4
      SET COLOR TO &c_normal
      @ 11, 0, 15, 15 BOX "┌┐│┘─└│"
      SET COLOR TO &c_unselected
      @ 12,1,14,14 BOX "         "
      @ 12, 1 SAY "O"
      @ 13, 2 SAY "Organization"
      @ 14, 5 SAY "Data"

CASE oldchoice =5
      SET COLOR TO &c_normal
      @ 11, 32, 15, 47 BOX "┌┐│┘─└│"
      SET COLOR TO &c_unselected
      @ 12,33,14,46 BOX "         "
      @ 12, 33 SAY "T"
      @ 13, 33 SAY "Authorizations"
      @ 14, 34 SAY "with Details"

CASE oldchoice =6
      SET COLOR TO &c_normal
      @ 11, 64, 15, 79 BOX "┌┐│┘─└│"
      SET COLOR TO &c_unselected
      @ 12,65,14,78 BOX "         "
      @ 12, 65 SAY "D"
      @ 13, 66 SAY "Discrepancies"
      @ 14, 66 SAY "Inspections"

CASE oldchoice =7
      SET COLOR TO &c_normal
      @ 16, 0, 20, 15 BOX "┌┐│┘─└│"
      SET COLOR TO &c_unselected
      @ 17,1,19,14 BOX "         "
      @ 17, 1 SAY "L"
      @ 18, 4 SAY "Reports"
      @ 19, 4 SAY "Listings"

CASE oldchoice =8
      SET COLOR TO &c_normal
      @ 16, 32, 20, 47 BOX "┌┐│┘─└│"
      SET COLOR TO &c_unselected
      @ 17,33,19,46 BOX "         "
      @ 17, 33 SAY "R  Read In"
      @ 18, 33 SAY "& Backup"
      @ 19, 33 SAY "Databases"

CASE oldchoice =9
```

```
            SET COLOR TO &c_normal
            @ 16, 64, 20, 79 BOX "┌┐ │┘─└│"
            SET COLOR TO &c_unselected
            @ 17,65,19,78 BOX "          "
            @ 17, 65 SAY "Q   Quit"
            @ 18, 65 SAY "and "
            @ 19, 65 SAY "Exit to DOS"

        ENDCASE
      ENDIF

    * end of keyhit loop
    ENDDO


    * perform selected option
    DO CASE

      CASE newchoice =1
        DO AUTHRZ

      CASE newchoice =2
        DO SUPPLY

      CASE newchoice =3
        DO IMBALANC

      CASE newchoice =4
        DO ORGANIZ

      CASE newchoice =5
        DO AUTHDOC

      CASE newchoice =6
        DO INSPECT

      CASE newchoice =7
        DO LISTINGS

      CASE newchoice =8
        DO READIN

      CASE newchoice =9
        DO CLEANUP

    ENDCASE

      * set old choice var to 0 so we get a highlight on the
    current option
      oldchoice =0

      * and set key input var to 0 so we don't fall out again
      key =0
```

```
ENDDO


***********
* PROC Setup.prg   define databases and index files
* Kevin M. Tanzer - 4 May 90
***********

* initialize databases and index files, along with alias
names
SELECT 1
  mLastArea = 1
  USE Basic INDEX BasicNSN AL  S Objectives
  GOTO TOP

SELECT 2
  mLastArea = 2
  USE OrgData INDEX OrgData ALIAS WhosWho
  GOTO TOP
  mLastOrg = ORG_NUMBR

SELECT 3
  mLastArea = 3
  USE Document INDEX Document ALIAS Details
  GOTO TOP

SELECT 4
  mLastArea = 4
  USE Inspect INDEX Inspect ALIAS Performance
  GOTO TOP

*** SELECT 5 - Area is used in Procedure AUTHDOC to build
temporary database
*** of supply documents to be viewed.

*** SELECT 9 - Area is used in Procedure READIN to read in
extracted SBSS
*** data into a long character string, which is then parsed
into the
*** fields of the BASIC.DBF (authorization) and DOCUMENT.DBF

RETURN
***
*** C:\CLIP\AUTHRZ.PRG : Add, Edit, Browse, Delete, Search
(standalone)
*** Generated on March 11, 1990
*** Source .WW file: C:\UI\WW\AUTHRZ.WW
*** Target environment: Clipper Summer 87
*** Modified by K. Tanzer on 11 Mar 90 for use with LIMP
V1.0

*** environment stuff
* environment
SET SCOREBOARD off
```

```
SET CONFIRM off
SET CURSOR ON
* menu initialization
PRIVATE key

* DBF initialization
SELECT 1
mLastArea = 1

*************
*
* main menu loop:
*
* iterates once for each time an option action is performed.
* this loop calls procedures to perform selected actions.
* (Procedures are defined below this loop)
*
*************

DO authscrn

DO WHILE .t.
help_code = 'authrz'

   * display entry record
   DO authdisp

   ***
   * user selects action here
   ***
   SET COLOR TO &c_unselected

   @ 24,2 PROMPT " Next "
   @ 24,9 PROMPT " Prev "
   @ 24,16 PROMPT " First "
   @ 24,24 PROMPT " Last "
   @ 24,38 PROMPT " Search "
   @ 24,47 PROMPT " Edit "
   @ 24,54 PROMPT " Append "
   @ 24,63 PROMPT " Delete "
   @ 24,72 PROMPT " Quit "

   MENU TO key

   * perform selected option
   DO CASE
      CASE key =1
           DO authnext
      CASE key =2
           DO authprev
      CASE key =3
           DO authfirst
      CASE key =4
           DO authlast
```

```
      CASE key =5
           DO authsearch
      CASE key =6
           DO authedit
      CASE key =7
           DO authappnd
      CASE key =8
           DO authdel
      CASE key =9
           DO authquit
           RETURN

   ENDCASE

ENDDO
RETURN
****************** END OF MENU PROCEDURE
************************************

**************
*
* action procedures:
*
* the following are called by the menu loop above.
*
**************


***********************************************
* authscrn:  displays the screen background
***********************************************
PROC authscrn
* display fixed text
CLEAR
SET COLOR TO &c_normal
@ 1, 0, 22, 79 BOX "┌┐│┘─└│"
SET COLOR TO &c_selected
@ 2,1,21,78 BOX "         "
@ 2, 30 SAY "Authorized WRM Items"
@ 3, 2 SAY "Item Description:"+space(22)+"NSN:"
@ 5, 2 SAY "Authorized Quantity:"+space(9)+"On Hand Qty:"
@ 7, 2 SAY "Unit of Issue:"+space(24)+"Document Number"
@ 8, 2 SAY "Price of Unit:"+space(24)+"of Authorization:"
@ 10, 2 SAY "Value of Units:"
@ 12, 2 SAY "ERRC Code:"
@ 24,1 SAY "|       |       |       |       |       |       |
   |       |       |       |"
@ 24,2 SAY " Next "
@ 24,9 SAY " Prev "
@ 24,16 SAY " First "
@ 24,24 SAY " Last "
@ 24,38 SAY " Search "
@ 24,47 SAY " Edit "
@ 24,54 SAY " Append "
```

```
@ 24,63 SAY " Delete "
@ 24,72 SAY " Quit "
RETURN


************************************
* authnext: go to next record
************************************
PROC authnext
SKIP
IF eof()
   GOTO BOTTOM
   DO statmsg with "End of file!"
ELSE
   DO statmsg with ""
   mLastOrg = SUBSTR(SOURCE_DOC,2,5)
   DO authdisp
ENDIF
RETURN


************************************
* authprev: go to previous record
************************************
PROC authprev
SKIP -1
IF bof()
   GOTO TOP
   DO statmsg with "Beginning of file!"
ELSE
   DO statmsg with ""
   mLastOrg = SUBSTR(SOURCE_DOC,2,5)
   DO authdisp
ENDIF
RETURN


************************************
* authfirst: go to first record
************************************
PROC authfirst
GOTO TOP
DO statmsg with ""
   mLastOrg = SUBSTR(SOURCE_DOC,2,5)
DO authdisp
RETURN


************************************
* authlast: go to last record
************************************
PROC authlast
GOTO BOTTOM
DO statmsg with ""
```

```
   mLastOrg = SUBSTR(SOURCE_DOC,2,5)
DO authdisp
RETURN


*****************************************************************
****************
* authsearch: search for a record, given entry into desired
criterion fields
*****************************************************************
****************
PROC authsearch
PRIV searchval,orecno
help_code = 'authsearch'
searchval = space(15)
DO statmsg with ""
@ 23,2 SAY "Enter NSN value: " GET searchval PICTURE '@!
NNNNNNNNNNNNNN'
READ
orecno = RECNO()
SEEK searchval
IF .NOT. FOUND()
   GOTO orecno
   DO statmsg with "NSN value not found!"
ELSE
   DO statmsg with ""
   orecno = recno()
   GOTO orecno
   mLastOrg = SUBSTR(SOURCE_DOC,2,5)
   DO authdisp
ENDIF
RETURN


**********************************
* authedit: edit current record
**********************************
PROC authedit

*** create memvar duplicates for all fields
PRIVATE mNOMENCLATR, mSTOCK_NBR, mATHRZD_QTY, mQTY_ONHAND,
mUNIT_ISSUE
PRIVATE mUNIT_PRICE, mSOURCE_DOC, mERRC_CODE
help_code = 'authedit'

*** ...and initialize 'em
mNOMENCLATR = NOMENCLATR
mSTOCK_NBR = STOCK_NBR
mATHRZD_QTY = ATHRZD_QTY
mQTY_ONHAND = QTY_ONHAND
mUNIT_ISSUE = UNIT_ISSUE
mUNIT_PRICE = UNIT_PRICE
```

```
mSOURCE_DOC = SOURCE_DOC
mERRC_CODE = ERRC_CODE

DO statmsg with "Edit record. ^W to save; Esc to abandon"

*** get input fields into memvar duplicates
@ 3, 20 GET mNOMENCLATR PICTURE '@! NNNNNNNNNNNNNNNNNN'
@ 3, 46 GET mSTOCK_NBR PICTURE '@! NNNNNNNNNNNNNN'
@ 5, 23 GET mATHRZD_QTY PICTURE '99999'
@ 5, 44 GET mQTY_ONHAND PICTURE '99999'
@ 7, 18 GET mUNIT_ISSUE PICTURE '@! NN'
@ 8, 18 GET mUNIT_PRICE PICTURE '9999999.99'
@ 9, 40 GET mSOURCE_DOC PICTURE '@! NNNNNNNNNNNNN'
@ 12, 13 GET mERRC_CODE PICTURE '@! NNN'
READ

DO statmsg with ""

IF lastkey() = 27       && user escaped out of READ
   RETURN
ENDIF


REPLACE NOMENCLATR WITH UPPER(mNOMENCLATR)
REPLACE STOCK_NBR WITH mSTOCK_NBR
REPLACE ATHRZD_QTY WITH mATHRZD_QTY
REPLACE QTY_ONHAND WITH mQTY_ONHAND
mTemp = mATHRZD_QTY - QTY_UNSUPP
REPLACE NET_QTY WITH mTemp
mTemp = mTemp - mQTY_ONHAND
REPLACE QTY_SHORT WITH mTemp
REPLACE UNIT_ISSUE WITH UPPER(mUNIT_ISSUE)
REPLACE UNIT_PRICE WITH mUNIT_PRICE
REPLACE SOURCE_DOC WITH UPPER(mSOURCE_DOC)
REPLACE ERRC_CODE WITH UPPER(mERRC_CODE)

   mLastOrg = SUBSTR(SOURCE_DOC,2,5)
DO authdisp
RETURN



**********************************
* authappnd: append new record
**********************************
PROC authappnd

DO ClassPop
* Popup reminder about classified data

DO authscrn
DO authdisp
*** create memvar duplicates for all fields
```

```
PRIVATE mNOMENCLATR, mSTOCK_NBR, mATHRZD_QTY, mQTY_ONHAND,
mUNIT_ISSUE
PRIVATE mUNIT_PRICE, mSOURCE_DOC, mERRC_CODE
help_code = 'authappnd'

*** ...and initialize 'em
mNOMENCLATR = SPACE(19)
mSTOCK_NBR = SPACE(15)
mATHRZD_QTY = 0
mQTY_ONHAND = 0
mUNIT_ISSUE = SPACE(2)
mUNIT_PRICE = 0.00
mSOURCE_DOC = SPACE(14)
mERRC_CODE = SPACE(3)

DO statmsg with "Enter new record. ^W to save; Esc to
abandon"

*** get input fields into memvar duplicates
@ 3, 20 GET mNOMENCLATR PICTURE '@! NNNNNNNNNNNNNNNNNN'
@ 3, 46 GET mSTOCK_NBR PICTURE '@! NNNNNNNNNNNNNNN'
@ 5, 23 GET mATHRZD_QTY PICTURE '99999'
@ 5, 44 GET mQTY_ONHAND PICTURE '99999'
@ 7, 18 GET mUNIT_ISSUE PICTURE '@! NN'
@ 8, 18 GET mUNIT_PRICE PICTURE '9999999.99'
@ 9, 40 GET mSOURCE_DOC PICTURE '@! NNNNNNNNNNNNNN'
@ 12, 13 GET mERRC_CODE PICTURE '@! NNN'
READ

DO statmsg with ""

IF lastkey() = 27       && user escaped out of READ
   RETURN
ENDIF

* append into DBF fields
APPEND BLANK
REPLACE NOMENCLATR WITH UPPER(mNOMENCLATR)
REPLACE STOCK_NBR WITH mSTOCK_NBR
REPLACE ATHRZD_QTY WITH mATHRZD_QTY
REPLACE QTY_ONHAND WITH mQTY_ONHAND
mTemp = ATHRZD_QTY - QTY_UNSUPP
REPLACE NET_QTY WITH mTemp
mTemp = mTemp - mQTY_ONHAND
REPLACE QTY_SHORT WITH mTemp
REPLACE UNIT_ISSUE WITH UPPER(mUNIT_ISSUE)
REPLACE UNIT_PRICE WITH mUNIT_PRICE
REPLACE SOURCE_DOC WITH UPPER(mSOURCE_DOC)
REPLACE ERRC_CODE WITH UPPER(mERRC_CODE)
mBACKUP = .T.

REINDEX

   mLastOrg = SUBSTR(SOURCE_DOC,2,5)
```

```
DO authdisp
RETURN


************************************************
* authdel: delete record. If deleted, recall
************************************************
PROC authdel
IF .not. deleted()
   DELETE
   mBASIC = .T.
ELSE
   RECALL
ENDIF
DO statmsg with ""
DO authdisp
RETURN


**********************************
* authquit: various quit things
**********************************
PROC authquit
SET COLOR TO &c_normal
SET CURSOR OFF
CLEAR
RETURN


************************************************
* statmsg: displays passed message on line 23
* this routine is common to all the subprograms
************************************************
PROC statmsg
PARAM s
@ 23,0 SAY " "+s+space(54-len(s))
RETURN


*****************************************************************
*********
* authdisp: displays current record, along with recno() and
deleted()
*****************************************************************
*********
PROC authdisp
PRIVATE mQTY_ONHAND

SELECT 1
mLastArea = 1
```

```
SET COLOR TO &c_selected
IF recno() = 0
   GOTO TOP
ENDIF

DO statmsg with ""

@ 3, 20 GET NOMENCLATR
@ 3, 46 GET STOCK_NBR
@ 5, 23 GET ATHRZD_QTY
mQTY_ONHAND = QTY_ONHAND
@ 5, 44 GET QTY_ONHAND
@ 7, 18 GET UNIT_ISSUE
@ 8, 18 GET UNIT_PRICE PICTURE '9999999.99'
NetValue = UNIT_PRICE * mQTY_ONHAND
@ 10, 18 SAY NetValue PICTURE '9999999.99'
@ 9, 40 GET SOURCE_DOC
@ 12, 13 GET ERRC_CODE
CLEAR GETS
@ 23,70 SAY iif(deleted(),"Deleted","       ")

RETURN
***
*** C:\CLIP\SUPPLY.PRG : Add, Edit, Browse, Delete, Search
(standalone)
*** Generated on March 11, 1990
*** Source .WW file: C:\UI\WW\SUPPLY.WW
*** Target environment: Clipper Summer 87
*** Modified by K. Tanzer on 11 Mar 90 for use with LIMP
V1.0

*** environment stuff
* environment
SET SCOREBOARD off
SET CONFIRM off
SET CURSOR ON

* menu initialization
PRIVATE key

* DBF initialization
* Open database DOCUMENT (alias DETAILS)
*
SELECT 3
mLastArea = 3
GOTO TOP

**********************************************************************
*************
*
* main menu loop:
*
* iterates once for each time an option action is performed.
* this loop calls procedures to perform selected actions.
```

```
* (Procedures are defined below this loop)
*
*************************************************************
*************

* display fixed text
SET COLOR TO &c_normal
CLEAR
SET COLOR TO &c_normal
@ 0, 0, 22, 79 BOX "┌┐│┘─└│"
SET COLOR TO &c_selected
@ 1,1,21,78 BOX "          "
@ 1, 25 SAY "Supply Document Details Screen"
@ 3, 5 SAY "Document Type:"+space(21)+"Ordering
Organization:"
@ 4, 3 SAY "Document Number:"+space(21)+"Date Ordered:"
@ 5, 39 SAY "Estimated Delivery Date:"
@ 6, 20 SAY "Status:"+space(15)+"Due In Doc:"
@ 9, 3 SAY "Item Ordered:"+space(30)+"Quantity Ordered:"
@ 10, 3 SAY "National Stock Number:"+space(27)+"Unit Price:"
@ 11, 5 SAY "Prime or Substitute:"+space(26)+"Total Value:"
SET COLOR TO &c_normal
@ 13, 4, 20, 55 BOX "┌┐│┘─└│"
SET COLOR TO &c_selected
@ 14,5,19,54 BOX "          "
@ 14, 15 SAY "Point of Contact Information"
@ 15, 6 SAY "Name:"
@ 16, 6 SAY "Organization:"
@ 17, 6 SAY "Office:"
@ 18, 6 SAY "Phone:"
@ 24,1 SAY "|       |       |       |       |       |         |
   |         |         |       |"
@ 24,2 SAY " Next "
@ 24,9 SAY " Prev "
@ 24,16 SAY " First "
@ 24,24 SAY " Last "
@ 24,38 SAY " Search "
@ 24,47 SAY " Edit "
@ 24,54 SAY " Append "
@ 24,63 SAY " Delete "
@ 24,72 SAY " Quit "

DO WHILE .t.
help_code = 'supply'
   * display entry record
   DO dispsupply


*************************************************************
***
   * user selects action here

*************************************************************
***
```

```
   SET COLOR TO &c_unselected

   @ 24,2 PROMPT " Next "
   @ 24,9 PROMPT " Prev "
   @ 24,16 PROMPT " First "
   @ 24,24 PROMPT " Last "
   @ 24,38 PROMPT " Search "
   @ 24,47 PROMPT " Edit "
   @ 24,54 PROMPT " Append "
   @ 24,63 PROMPT " Delete "
   @ 24,72 PROMPT " Quit "

   MENU TO key

   * perform selected option
   DO CASE
      CASE key =1
           DO nextsupply
      CASE key =2
           DO prevsupply
      CASE key =3
           DO firstsuppl
      CASE key =4
           DO lastsupply
      CASE key =5
           DO srchsupply
      CASE key =6
           DO editsupply
      CASE key =7
           DO appsupply
      CASE key =8
           DO delsupply
      CASE key =9
           DO quitsupply
           RETURN

   ENDCASE

ENDDO
RETURN
********** END OF MENU PROCEDURE
**********************************

*************************************************************************
**************
*
* action procedures:
*
* the following are called by the menu loop above.
*
*************************************************************************
**************
```

```
************************************
* nextsupply: go to next record
************************************
PROC nextsupply
SKIP
IF eof()
   GOTO BOTTOM
   DO statmsg with "End of file!"
ELSE
   DO statmsg with ""
   mLastOrg = SUBSTR(DOC_NUMBER,2,5)
   DO dispsupply
ENDIF
RETURN


************************************
* prevsupply: go to previous record
************************************
PROC prevsupply
SKIP -1
IF bof()
   GOTO TOP
   DO statmsg with "Beginning of file!"
ELSE
   DO statmsg with ""
   mLastOrg = SUBSTR(DOC_NUMBER,2,5)
   DO dispsupply
ENDIF
RETURN


************************************
* firstsuppl: go to first record
************************************
PROC firstsuppl
GOTO TOP
DO statmsg with "Top of file!"
   mLastOrg = SUBSTR(DOC_NUMBER,2,5)
DO dispsupply
RETURN


************************************
* lastsupply: go to last record
************************************
PROC lastsupply
GOTO BOTTOM
DO statmsg with "Bottom of file"
   mLastOrg = SUBSTR(DOC_NUMBER,2,5)
DO dispsupply
RETURN
```

```
*****************************************************************
*****************
* srchsupply: search for a record, given entry into desired
criterion fields
*****************************************************************
*****************
PROC srchsupply
PRIV searchval,orecno
help_code ='srchsupply'
searchval = space(14)
DO statmsg with ""
@ 23,2 SAY "Enter Document number: " GET searchval PICTURE
'@! NNNNNNNNNNNNNN'
READ
orecno = recno()
SEEK searchval
IF .NOT. FOUND()
   GOTO orecno
   DO statmsg with "Search value not found!"
ELSE
   DO statmsg with ""
   mLastOrg = SUBSTR(DOC_NUMBER,2,5)
   DO dispsupply
ENDIF
RETURN



*********************************
* editsupply: edit current record
*********************************
PROC editsupply

*** create memvar duplicates for all fields
PRIVATE
mDOC_NUMBER,mSTOCK_NBR,mPRIME_SUB,mEDD,mSTATUS,mDUE_IN_DOC,m
QTY_ORDERD
help_code = 'editsupply'

*** ...and initialize 'em
mDOC_NUMBER = DOC_NUMBER
mSTOCK_NBR = STOCK_NBR
mPRIME_SUB = PRIME_SUB
mEDD = EDD
mSTATUS = STATUS
mDUE_IN_DOC = DUE_IN_DOC
mQTY_ORDERD = QTY_ORDERD
DO statmsg with "Edit record. ^W to save; Esc to abandon"

*** get input fields into memvar duplicates
@ 4, 20 GET mDOC_NUMBER PICTURE '@! NNNNNNNNNNNNN'
@ 5, 65 GET mEDD PICTURE '9999'
@ 6, 28 GET mSTATUS PICTURE '@! AAAAA'
```

```
@ 6, 54 GET mDUE_IN_DOC PICTURE '@! NNNNNNNNNNNNNN'
@ 9, 64 GET mQTY_ORDERD PICTURE '99999'
@ 10, 26 GET mSTOCK_NBR PICTURE '@! NNNNNNNNNNNNNNN'
@ 11, 26 GET mPRIME_SUB PICTURE '@! A'
READ

DO statmsg with ""

IF lastkey() = 27        && user escaped out of READ
   RETURN
ENDIF

REPLACE DOC_NUMBER WITH UPPER(mDOC_NUMBER)
REPLACE STOCK_NBR WITH UPPER(mSTOCK_NBR)
REPLACE PRIME_SUB WITH UPPER(mPRIME_SUB)
REPLACE EDD WITH mEDD
REPLACE STATUS WITH UPPER(mSTATUS)
REPLACE DUE_IN_DOC WITH UPPER(mDUE_IN_DOC)
REPLACE QTY_ORDERD WITH mQTY_ORDERD

   mLastOrg = SUBSTR(DOC_NUMBER,2,5)
DO dispsupply

RETURN


**********************************
* appsupply: append new record
**********************************
PROC appsupply

*** create memvar duplicates for all fields
PRIVATE
mDOC_NUMBER,mSTOCK_NBR,mPRIME_SUB,mEDD,mSTATUS,mDUE_IN_DOC,m
QTY_ORDERD
help_code = 'appsupply'

*** ...and initialize 'em
mDOC_NUMBER = SPACE(14)
mSTOCK_NBR = SPACE(15)
mPRIME_SUB = SPACE(1)
mEDD = 0
mSTATUS = "BO"
mDUE_IN_DOC = SPACE(14)
mQTY_ORDERD = 0

DO statmsg with "Enter new record. ^W to save; Esc to
abandon"

*** get input fields into memvar duplicates
@ 4, 20 GET mDOC_NUMBER PICTURE '@! NNNNNNNNNNNNNN'
@ 5, 65 GET mEDD PICTURE '9999'
@ 6, 28 GET mSTATUS PICTURE '@! AAAAAA'
@ 6, 54 GET mDUE_IN_DOC PICTURE '@! NNNNNNNNNNNNN'
```

```
@ 9, 64 GET mQTY_ORDERD PICTURE '99999'
@ 10, 26 GET mSTOCK_NBR PICTURE '@! NNNNNNNNNNNNNNN'
@ 11, 26 GET mPRIME_SUB PICTURE '@! A'

READ

DO statmsg with ""

IF lastkey() = 27        && user escaped out of READ
   RETURN
ENDIF

* append into DBF fields
APPEND BLANK
REPLACE DOC_NUMBER WITH UPPER(mDOC_NUMBER)
REPLACE STOCK_NBR WITH UPPER(mSTOCK_NBR)
     mSTOCK_NBR = UPPER(mSTOCK_NBR)
REPLACE PRIME_SUB WITH UPPER(mPRIME_SUB)
REPLACE EDD WITH mEDD
REPLACE STATUS WITH UPPER(mSTATUS)
REPLACE DUE_IN_DOC WITH UPPER(mDUE_IN_DOC)
REPLACE QTY_ORDERD WITH mQTY_ORDERD
REPLACE DOC_TYPE WITH 'O'
mBACKUP = .T.

REINDEX
   mLastOrg = SUBSTR(DOC_NUMBER,2,5)
DO dispsupply
RETURN



************************************************
* delsupply: delete record. If deleted, recall
************************************************
PROC delsupply
IF .not. deleted()
   DELETE
   mDOCUMENT = .T.
ELSE
   RECALL
ENDIF
DO statmsg with ""
DO dispsupply
RETURN


**********************************
* quitsupply: various quit things
**********************************
PROC quitsupply
SET COLOR TO &c_normal
SET CURSOR OFF
CLEAR
```

```
RETURN


*****************************************************************
*********
* dispsupply: displays current record, along with recno()
and deleted()
*****************************************************************
*********
PROC dispsupply
PRIVATE
mORG,mDOC,mDATE,mITEM,mQUANT,mPRICE,mVALUE,mTYPE,mSTOCK_NBR,
searchval,orecno

SELECT 3
mLastArea = 3
IF recno() = 0
   GOTO 1
   mLastOrg = SUBSTR(DOC_NUMBER,2,5)
ENDIF

mDOC = DOC_TYPE
DO Doc_Look with mDOC
*** Translate the DOC_TYPE to a plain english string
@ 3, 20 SAY mDOC

@ 4, 20 GET DOC_NUMBER
*** Break Document Number into separate elements
mORG = SUBSTR(DOC_NUMBER,2,5)
mDATE = SUBSTR(DOC_NUMBER,7,4)
@ 3, 64 SAY mORG
@ 4, 55 SAY mDATE

mSTOCK_NBR = STOCK_NBR
SELECT 1
mLastArea = 1
LOCATE ALL FOR STOCK_NBR = mSTOCK_NBR
@ 9, 17 SAY NOMENCLATR
mPRICE = UNIT_PRICE

*** Lookup POC Information based on organization number
SELECT 2
mLastArea = 2
searchval = mORG
orecno = recno()
SEEK searchval
IF .NOT. FOUND() .OR. OFFICE = 'UNKNOWN'
   GOTO orecno
   DO statmsg with "This Org/Shop number needs to be
defined."
ELSE
   @ 15, 12 GET POC_NAME
   @ 16, 20 GET SQUADRON
```

```
   @ 17, 15 GET OFFICE
   @ 18, 14 GET POC_PHONE PICTURE '@R 999-9999'
ENDIF

SELECT 3
mLastArea =3
@ 5, 65 GET EDD PICTURE '9999'
@ 6, 28 GET STATUS
@ 6, 54 GET DUE_IN_DOC
@ 9, 64 GET QTY_ORDERD
@ 10, 64 SAY mPRICE PICTURE '9999999.99'
@ 10, 26 GET STOCK_NBR
@ 11, 26 GET PRIME_SUB
mPRICE = mPRICE*QTY_ORDERD
@ 11, 64 SAY mPRICE PICTURE '9999999.99'

CLEAR GETS
@ 23,70 SAY iif(deleted(),"Deleted","       ")

RETURN
***
*** C:\CLIP\IMBALANC.PRG : Add, Edit, Browse, Delete, Search
(standalone)
*** Generated on March 11, 1990
*** Source .WW file: C:\UI\WW\IMBALANC.WW
*** Target environment: Clipper Summer 87
*** Modified by K. Tanzer on 11 Mar 90 for use with LIMP
V1.0

*** environment stuff
* environment
SET SCOREBOARD off
SET CONFIRM off
SET CURSOR ON

* menu initialization
PRIVATE key


* DBF initialization
* Use Basic database
SELECT 1
mLastArea = 1
*** SET FILTER TO QTY_SHORT <> 0
IF EOF()
  GOTO TOP
ENDIF

**************************************************************
**************
*
* main menu loop:
*
* iterates once for each time an option action is performed.
```

```
* this loop calls procedures to perform selected actions.
* (Procedures are defined below this loop)
*
*****************************************************************
*************

* display fixed text
SET COLOR TO &c_normal
CLEAR
SET COLOR TO &c_normal
@ 0, 0, 22, 79 BOX "┌┐│┘─└│"
SET COLOR TO &c_selected
@ 1,1,21,78 BOX "         "
@ 1, 18 SAY "Authorized vs On-hand Quantities"
@ 4, 4 SAY "Item Description:"+space(27)+"NSN:"
@ 9, 4 SAY "Total Quantity Authorized:"
@ 11, 4 SAY "Authorized Unsupportable:"
@ 13, 9 SAY "Net Quantity Authorized:"
@ 15, 10 SAY "Total Quantity On-hand:"
@ 17, 18 SAY "Quantity Short:"
@ 24,1 SAY "|       |       |       |       |       |       |
   |       |       |       |"
@ 24,2 SAY " Next "
@ 24,9 SAY " Prev "
@ 24,16 SAY " First "
@ 24,24 SAY " Last "
@ 24,38 SAY " Search "
@ 24,47 SAY " Edit "
@ 24,54 SAY " Append "
@ 24,63 SAY " Delete "
@ 24,72 SAY " Quit "

DO WHILE .t.
help_code = 'imbalanc'

   * display entry record
   DO imbaldisp


*****************************************************************
***
   * user selects action here

*****************************************************************
***
   SET COLOR TO &c_unselected

   @ 24,2 PROMPT " Next "
   @ 24,9 PROMPT " Prev "
   @ 24,16 PROMPT " First "
   @ 24,24 PROMPT " Last "
   @ 24,38 PROMPT " Search "
   @ 24,47 PROMPT " Edit "
   @ 24,54 PROMPT " Append "
```

```
   @ 24,63 PROMPT " Delete "
   @ 24,72 PROMPT " Quit "

   MENU TO key

   * perform selected option
   DO CASE
      CASE key =1
           DO imbalnext
      CASE key =2
           DO imbalprev
      CASE key =3
           DO imbalfirst
      CASE key =4
           DO imballast
      CASE key =5
           DO imbalsrch
      CASE key =6
           DO imbaledit
      CASE key =7
           DO imbalappnd
      CASE key =8
           DO imbaldel
      CASE key =9
           DO imbalquit
           RETURN
   ENDCASE
ENDDO
RETURN
****************** END OF MENU PROCEDURE
**********************************

*************************************************************************
**************
*
* action procedures:
*
* the following are called by the menu loop above.
*
*************************************************************************
**************


**********************************
* imbalnext: go to next record
**********************************
* Will locate the next record that has a non-zero
* shortage quantity
PROC imbalnext
SKIP
IF eof()
   GOTO BOTTOM
   DO statmsg with "End of File!"
ELSE
```

```
   DO statmsg with ""
   mLastOrg = SUBSTR(SOURCE_DOC,2,5)
   DO imbaldisp
ENDIF
RETURN


************************************
* imbalprev: go to previous record
************************************
PROC imbalprev
SKIP -1
IF bof()
   GOTO TOP
   DO statmsg with "Beginning of File!"
ELSE
   mLastOrg = SUBSTR(SOURCE_DOC,2,5)
   DO statmsg with ""
ENDIF
DO imbaldisp
RETURN


************************************
* imbalfirst: go to first record
************************************
PROC imbalfirst
GOTO TOP
DO statmsg with ""
   mLastOrg = SUBSTR(SOURCE_DOC,2,5)
DO imbaldisp
RETURN


************************************
* imballast: go to last record
************************************
PROC imballast
GOTO BOTTOM
DO statmsg with ""
   mLastOrg = SUBSTR(SOURCE_DOC,2,5)
DO imbaldisp
RETURN



************************************************************************
****************
* imbalsrch: search for a record, given entry into desired
criterion fields
************************************************************************
****************
PROC imbalsrch
PRIV searchval,orecno
```

```
help_code = 'imbalsrch'
searchval = space(15)
DO statmsg with ""
* Make a search by National Stock Number
@ 23,2 SAY "Enter NSN search value: " GET searchval PICTURE
'@! NNNNNNNNNNNNNNN'
READ
orecno = recno()
SEEK searchval
IF .NOT. FOUND()
   GOTO orecno
   TONE(900,2)
   DO statmsg with "That NSN not found!"
ELSE
   DO statmsg with ""
   mLastOrg = SUBSTR(SOURCE_DOC,2,5)
   DO imbaldisp
ENDIF
RETURN



**********************************
* imbaledit: edit current record
**********************************
PROC imbaledit

*** create memvar duplicates for all fields
PRIVATE mNOMENCLATR, mSTOCK_NBR, mATHRZD_QTY, mQTY_UNSUPP,
mNET_QTY
PRIVATE mQTY_ONHAND, mQTY_SHORT
help_code = 'imbaledit'
*** ...and initialize 'em
mNOMENCLATR = NOMENCLATR
mSTOCK_NBR = STOCK_NBR
mATHRZD_QTY = ATHRZD_QTY
mQTY_UNSUPP = QTY_UNSUPP
mNET_QTY = mATHRZD_QTY - mQTY_UNSUPP
mQTY_ONHAND = QTY_ONHAND
mQTY_SHORT = mNET_QTY - mQTY_ONHAND

DO statmsg with "Edit record. ^W to save; Esc to abandon"

*** get input fields into memvar duplicates
@ 4, 22 GET mNOMENCLATR PICTURE '@! NNNNNNNNNNNNNNNNNN'
@ 4, 53 GET mSTOCK_NBR PICTURE '@! NNNNNNNNNNNNNN'
@ 9, 31 GET mATHRZD_QTY PICTURE '99999'
@ 11, 31 GET mQTY_UNSUPP PICTURE '99999'
@ 13, 34 GET mNET_QTY  PICTURE '99999'
@ 15, 34 GET mQTY_ONHAND PICTURE '99999'
@ 17, 34 GET mQTY_SHORT PICTURE '99999'
READ

DO statmsg with ""
```

```
IF lastkey() = 27        && user escaped out of READ
   RETURN
ENDIF

REPLACE NOMENCLATR WITH UPPER(mNOMENCLATR)
REPLACE STOCK_NBR WITH mSTOCK_NBR
REPLACE ATHRZD_QTY WITH mATHRZD_QTY
REPLACE QTY_UNSUPP WITH mQTY_UNSUPP
REPLACE NET_QTY WITH mNET_QTY
REPLACE QTY_ONHAND WITH mQTY_ONHAND
REPLACE QTY_SHORT WITH mQTY_SHORT

DO imbaldisp
RETURN



**********************************
* imbalappnd: append new record
**********************************
PROC imbalappnd

*** create memvar duplicates for all fields
PRIVATE mNOMENCLATR, mSTOCK_NBR, mATHRZD_QTY, mQTY_UNSUPP,
mNET_QTY
PRIVATE mQTY_ONHAND, mQTY_SHORT
help_code = 'imbalappnd'

*** ...and initialize 'em
mNOMENCLATR = SPACE(19)
mSTOCK_NBR = SPACE(15)
mATHRZD_QTY = 00000
mQTY_UNSUPP = 00000
mNET_QTY = 00000
mQTY_ONHAND = 00000
mQTY_SHORT = 00000

DO statmsg with "Enter new record. ^W to save; Esc to
abandon"

*** get input fields into memvar duplicates
@ 4, 22 GET mNOMENCLATR PICTURE '@! NNNNNNNNNNNNNNNNNN'
@ 4, 53 GET mSTOCK_NBR PICTURE '@! NNNNNNNNNNNNNNN'
@ 9, 31 GET mATHRZD_QTY PICTURE '99999'
@ 11, 31 GET mQTY_UNSUPP PICTURE '99999'
@ 13, 34 GET mNET_QTY PICTURE '99999'
@ 15, 34 GET mQTY_ONHAND PICTURE '99999'
@ 17, 34 GET mQTY_SHORT PICTURE '99999'
READ

DO statmsg with ""

IF lastkey() = 27        && user escaped out of READ
```

```
   RETURN
ENDIF

* append into DBF fields
APPEND BLANK
REPLACE NOMENCLATR WITH UPPER(mNOMENCLATR)
REPLACE STOCK_NBR WITH mSTOCK_NBR
REPLACE ATHRZD_QTY WITH mATHRZD_QTY
REPLACE QTY_UNSUPP WITH mQTY_UNSUPP
REPLACE NET_QTY WITH mNET_QTY
REPLACE QTY_ONHAND WITH mQTY_ONHAND
REPLACE QTY_SHORT WITH mQTY_SHORT
mBACKUP = .T.

REINDEX

   mLastOrg = SUBSTR(SOURCE_DOC,2,5)
DO imbaldisp
RETURN



************************************************
* imbaldel: delete record. If deleted, recall
************************************************
PROC imbaldel
IF .not. deleted()
   DELETE
   mBASIC = .T.
ELSE
   RECALL
ENDIF
DO statmsg with ""
DO imbaldisp
RETURN


**********************************
* imbalquit: various quit things
**********************************
PROC imbalquit
SET COLOR TO &c_normal
SET FILTER TO
SET CURSOR OFF
CLEAR
RETURN



*****************************************************************
*********
* imbaldisp: displays current record, along with recno() and
deleted()
```

```
*****************************************************************
*********
PROC imbaldisp
IF recno() = 0
   GOTO 1
ENDIF
@ 4, 22 GET NOMENCLATR
@ 4, 53 GET STOCK_NBR
@ 9, 31 GET ATHRZD_QTY
@ 11, 31 GET QTY_UNSUPP
@ 13, 34 GET NET_QTY
@ 15, 34 GET QTY_ONHAND
@ 17, 34 GET QTY_SHORT
CLEAR GETS
@ 23,70 SAY iif(deleted(),"Deleted","       ")

RETURN
*****************************************************************
***********
* PROC Setup.prg   define databases and index files
* Kevin M. Tanzer - 4 May 90
*****************************************************************
***********

* initialize databases and index files, along with alias
names
SELECT 1
  mLastArea = 1
  USE Basic INDEX BasicNSN ALIAS Objectives
  GOTO TOP

SELECT 2
  mLastArea = 2
  USE OrgData INDEX OrgData ALIAS WhosWho
  GOTO TOP
  mLastOrg = ORG_NUMBR

SELECT 3
  mLastArea = 3
  USE Document INDEX Document ALIAS Details
  GOTO TOP

SELECT 4
  mLastArea = 4
  USE Inspect INDEX Inspect ALIAS Performance
  GOTO TOP

*** SELECT 5 - Area is used in Procedure AUTHDOC to build
temporary database
*** of supply documents to be viewed.

*** SELECT 9 - Area is used in Procedure READIN to read in
extracted SBSS
```

```
*** data into a long character string, which is then parsed
into the
*** fields of the BASIC.DBF (authorization) and DOCUMENT.DBF

RETURN

***
*** C:\CLIPPER\AUTHDOC.PRG : Add, Edit, Browse, Delete,
Search
*** also with scrolling lookup window for details
*** Generated on April 7, 1990
*** Source .WW file: C:\UI\WW\AUTHDOC2.WW
*** Target environment: Clipper Summer 87
*** Modified by K. Tanzer on 11 Mar 90 for use with LIMP
V1.0

* menu initialization
PRIVATE key,mKEY
SET CURSOR ON

* DBF initialization
SELECT 1
mLastArea = 1
GOTO TOP

**********************************************************************
*************
*
* main menu loop:
*
* iterates once for each time an option action is performed.
* this loop calls procedures to perform selected actions.
* (Procedures are defined below this loop)
*
**********************************************************************
*************

* display fixed text
SET COLOR TO &c_normal
CLEAR
SET COLOR TO &c_normal
@ 0, 0, 9, 79 BOX '┌─┐│┘─└│'
SET COLOR TO &c_selected
@ 1,1,8,78 BOX '          '
@ 1, 23 SAY 'Authorizations and Supply Details'
@ 2, 2 SAY 'Prime'+space(23)+'National'
@ 3, 2 SAY
'Authorized'+space(18)+'Stock'+space(20)+'Document'
@ 4, 2 SAY 'Item:'+space(23)+'Number:'+space(18)+'Number:'
@ 6, 2 SAY 'Authorized Quantity:'
@ 8, 6 SAY 'Quantity Onhand:'
@ 24,1 SAY '|       |       |       |       |       |
                |       |'
@ 24,2 SAY ' Next '
```

```
@ 24,9 SAY ' Prev '
@ 24,16 SAY ' First '
@ 24,24 SAY ' Last '
@ 24,32 SAY ' Search '
@ 24,42 SAY ' View Supply Details '
@ 24,72 SAY ' Quit '

DO WHILE .t.
help_code = 'authdoc'

   * display entry record
   DO disprec


*****************************************************************
***
   * user selects action here

*****************************************************************
***
   SET COLOR TO &c_unselected

   @ 24,2 PROMPT ' Next '
   @ 24,9 PROMPT ' Prev '
   @ 24,16 PROMPT ' First '
   @ 24,24 PROMPT ' Last '
   @ 24,32 PROMPT ' Search '
   @ 24,42 PROMPT ' View Supply Details '
   @ 24,72 PROMPT ' Quit '

   MENU TO key

   * perform selected option
   DO CASE
      CASE key =1
           DO nextthing
      CASE key =2
           DO prevthing
      CASE key =3
           DO firstthing
      CASE key =4
           DO lastthing
      CASE key =5
           DO searchthing
      CASE key =6
           mKEY = STOCK_NBR
           DO details with mKEY
      CASE key =7
           DO quitthing
          . RETURN

   ENDCASE

ENDDO
```

```
*****************************************************************
**************
*
* action procedures:
*
* the following are called by the menu loop above.
*
*****************************************************************
**************


***********************************
* nextrec: go to next record
***********************************
PROC nextthing
SKIP
IF eof()
   GOTO BOTTOM
   DO statmsg with 'End of file!'
ELSE
   DO statmsg with ''
   mLastOrg = SUBSTR(SOURCE_DOC,2,5)
   DO disprec
ENDIF
RETURN


***********************************
* prevrec: go to previous record
***********************************
PROC prevthing
SKIP -1
IF bof()
   GOTO TOP
   DO statmsg with 'Beginning of file!'
ELSE
   DO statmsg with ''
   mLastOrg = SUBSTR(SOURCE_DOC,2,5)
   DO disprec
ENDIF
RETURN


***********************************
* firstrec: go to first record
***********************************
PROC firstthing
GOTO TOP
DO statmsg with ''
   mLastOrg = SUBSTR(SOURCE_DOC,2,5)
DO disprec
RETURN
```

```
**********************************
* lastrec: go to last record
**********************************
PROC lastthing
GOTO BOTTOM
DO statmsg with ''
   mLastOrg = SUBSTR(SOURCE_DOC,2,5)
DO disprec
RETURN


**********************************************************************
*****************
* search: search for a record, given entry into desired
criterion fields
**********************************************************************
*****************
PROC searchthing
PRIV searchval,orecno
help_code = 'searchthng'
searchval = space(14)
DO statmsg with ''
@ 23,2 SAY 'Enter Document number: ' GET searchval PICTURE
'@! NNNNNNNNNNNNNN'
READ
orecno = recno()
LOCATE ALL FOR SOURCE_DOC = searchval
IF .NOT. FOUND()
   GOTO orecno
   DO statmsg with 'Document number not found!'
   mLastOrg = SUBSTR(SOURCE_DOC,2,5)
ELSE
   DO statmsg with ''
   mLastOrg = SUBSTR(SOURCE_DOC,2,5)
   DO disprec
ENDIF
RETURN


**********************************
* quitstuff: various quit things
**********************************
PROC quitthing
SET COLOR TO &c_normal
SET CURSOR OFF
CLEAR
RETURN


**********************************************************************
*********
```

```
* disprec: displays current record, along with recno() and
deleted()
*****************************************************************
*********
PROC disprec

@ 4, 8 GET NOMENCLATR
@ 4, 38 GET STOCK_NBR
@ 4, 63 GET SOURCE_DOC
@ 6, 23 GET NET_QTY
@ 8, 23 GET QTY_ONHAND
CLEAR GETS
@ 23,70 SAY iif(deleted(),'Deleted','       ')

RETURN



*****************************************************************
*********
* Details: scrolling lookup window for supply details
*****************************************************************
*********
* Creates a scrolling window to view all documents based on
the
* National Stock Number of the Authorization being viewed.
PROC Details
PARAM extract_key

PRIVATE  up, down, left, right, pgup, pgdn, home, end,
car_ret, ;
scrltop, scrlleft, scrlbot, scrlright, scrlheight, ;
currow, lastrec, firstrec, saverec, oldrec, xkey
help_code = 'details'

* inkey aliases
up = 5
down = 24
left = 19
right = 4
pgup = 18
pgdn = 3
home = 1
end = 6
car_ret = 13

* scroll area coords & dimensions (does not include box
border/header)
scrltop = 12
scrlleft = 1
scrlbot = 20
scrlright = 78
scrlheight = scrlbot - scrltop +1
currow = scrltop
```

```
oldrec = recno()

SET CURSOR OFF

DO statmsg with 'Collecting documents, one moment please...'
SELECT 3
mLastArea = 3
COPY TO TEMP2 FIELDS
DOC_NUMBER,DOC_TYPE,STOCK_NBR,QTY_ORDERD, ;
 DUE_IN_DOC,EDD FOR STOCK_NBR = extract_key
SELECT 5
mLastArea = 5
USE TEMP2
* Copies only the needed records to a temporary database,
this is ok because
* the user can _only_ view the records

GOTO TOP
firstrec = recno()
GOTO BOTTOM
lastrec = recno()


* pop the lookup window
Details = savescreen(10,0,21,79)
SET COLOR TO &c_normal
@ 10, 0, 21, 79 BOX '┌┐│┘─└│'
@ 10, 20 SAY ' Details supporting this authorization '
SET COLOR TO &c_unselected
@ 11,1,20,78 BOX '         '
@ 11, 2 SAY 'Document #     Type  NSN'+space(13)+'Date
Ordered  Date Due  Quantity'

DO statmsg with 'Use ^X^Y to scroll documents, <Esc> to
exit'

* display first window of records, hilite first one
GOTO TOP
DO De_disp WITH .t.
DO De_hidsp

xkey = 0

DO WHILE xkey <> 27 .and. xkey <> car_ret  && not <ESC> or
<CR>
   xkey = inkey(0)
   skey = upper(chr(xkey))

   DO CASE

      CASE xkey = up
           IF recno() = firstrec
              tone(900,2)
```

```
            ELSE
               IF currow > scrltop
                  DO De_lodsp
                  currow = currow - 1
                  SKIP -1
                  DO De_hidsp
               ELSE
                  DO De_lodsp
                  SKIP -1
                  scroll(scrltop, scrlleft, scrlbot,
scrlright, -1)
                  DO De_hidsp
               ENDIF
            ENDIF

      CASE xkey = down
            IF recno() = lastrec
               tone(900,2)
            ELSE
               IF currow < scrlbot
                  DO De_lodsp
                  currow = currow + 1
                  SKIP
                  DO De_hidsp
               ELSE
                  DO De_lodsp
                  SKIP
                  scroll(scrltop, scrlleft, scrlbot,
scrlright, 1)
                  DO De_hidsp
               ENDIF
            ENDIF

      CASE xkey = pgdn
            SKIP scrltop - currow + (2*scrlheight) -1
            IF eof()
               tone(900,2)
               SKIP -scrlheight
               DO De_disp WITH .f.
               DO De_hidsp
            ELSE
               SKIP 1 -scrlheight
               DO De_disp WITH .t.
               DO De_hidsp
            ENDIF

      CASE xkey = pgup
            SKIP scrltop - currow - scrlheight
            IF bof()
               tone(900,2)
               GOTO TOP
               DO De_disp WITH .t.
               DO De_hidsp
            ELSE
```

```
               DO De_disp WITH .t.
               DO De_hidsp
            ENDIF

      CASE xkey = home
            GOTO TOP
            DO De_disp WITH .t.
            DO De_hidsp

      CASE xkey = end
            GOTO BOTT
            SKIP 1 - scrlheight
            IF bof()
               GOTO TOP
            ENDIF
            DO De_disp WITH .f.
            DO De_hidsp

   ENDCASE
ENDDO

DO statmsg with ''
restscreen(10,0,21,79,Details)
SET COLOR TO &c_unselected

*** SELECT 5   && not really neccesary, just in case
though....
*** mLastArea = 5
*** ZAP              && clean out temporary database
SELECT 1
mLastArea = 1
GOTO oldrec
RETURN              && go back to bottom line menu


***
*** De_hidsp: hilites prompt line at currow
***
PROC De_hidsp
SET COLOR TO &c_selected
@ currow, 2 SAY DOC_NUMBER
@ currow, 17 SAY DOC_TYPE
@ currow, 23 SAY STOCK_NBR
@ currow, 43 SAY SUBSTR(DOC_NUMBER,7,4)
@ currow, 57 SAY EDD
@ currow, 63 SAY QTY_ORDERD
RETURN

***
*** De_lodsp: lolites prompt line at currow
***
PROC De_lodsp
SET COLOR TO &c_unselected
@ currow, 2 SAY DOC_NUMBER
```

```
@ currow, 17 SAY DOC_TYPE
@ currow, 23 SAY STOCK_NBR
@ currow, 43 SAY SUBSTR(DOC_NUMBER,7,4)
@ currow, 57 SAY EDD
@ currow, 63 SAY QTY_ORDERD
RETURN


***
*** De_disp : displays a window full of records.
***
*** If called WITH .t., currow and recno() will be set to
the first
*** record in the window.
*** If called WITH .f., currow and recno() will be set to
the last
*** record in the window.
***
PROC De_disp
PARAM set_to_top

PRIVATE thisrow, toprec

SET COLOR TO &c_unselected
@ scrltop, scrlleft CLEAR TO scrlbot, scrlright
@ 11, 2 SAY 'Document #     Type  NSN'+space(13)+'Date
Ordered  Date Due  Quantity'

toprec = recno()
thisrow = scrltop
DO WHILE .not. eof() .and. thisrow <= scrlbot
   @ thisrow, 2 SAY DOC_NUMBER
   @ thisrow, 17 SAY DOC_TYPE
   @ thisrow, 23 SAY STOCK_NBR
   @ thisrow, 43 SAY SUBSTR(DOC_NUMBER,7,4)
   @ thisrow, 57 SAY EDD
   @ thisrow, 63 SAY QTY_ORDERD
   SKIP
   thisrow = thisrow + 1
ENDDO

IF set_to_top
   GOTO toprec
   currow = scrltop
ELSE
   SKIP -1
   currow = thisrow -1
ENDIF

RETURN
***
*** C:\CLIP\INSPECT.PRG : Add, Edit, Browse, Delete, Search
(standalone)
*** Generated on March 11, 1990
```

```
*** Source .WW file: C:\UI\WW\INSPECT.WW
*** Target environment: Clipper Summer 87
*** Modified by K. Tanzer on 11 Mar 90 for use with LIMP
V1.0

*** environment stuff
* environment
SET SCOREBOARD off
SET CONFIRM off
SET CURSOR ON

* menu initialization
PRIVATE key

* DBF initialization
* Use database INSPECT
*
SELECT 4
mLastArea = 4
GOTO TOP
LOCATE FOR ORG_NUMBER = mLastOrg
IF .NOT. FOUND()
  GOTO TOP
ENDIF
**********************************************************************
*************
*
* main menu loop:
*
* iterates once for each time an option action is performed.
* this loop calls procedures to perform selected actions.
* (Procedures are defined below this loop)
*
**********************************************************************
*************

* display fixed text
SET COLOR TO &c_normal
CLEAR
SET COLOR TO &c_normal
@ 0, 0, 22, 79 BOX '┌┐│┘─└│'
SET COLOR TO &c_selected
@ 1,1,21,78 BOX '          '
@ 1, 21 SAY 'Inspection and Discrepancy Information'
@ 3, 2 SAY 'Wing:'+space(33)+'Last Inspection:'
@ 4, 2 SAY 'Squadron:'+space(29)+'Next Inspection Due:'
@ 5, 2 SAY 'Branch:'
@ 6, 2 SAY 'Office Symbol:'
@ 6, 38 SAY 'Org Number:'
@ 8, 15 SAY 'Inspection Results'
@ 9, 2 SAY 'Date:'
@ 10, 2 SAY 'Discrepancy:'
@ 11, 2 SAY 'Reference:'
@ 12, 2 SAY 'Response:'
```

```
@ 13, 2 SAY 'Follow Up (Y/N):     Reinspection on:'
@ 14, 2 SAY 'STATUS:'
@ 24,1 SAY '|       |       |       |       |       |       |
   |          |       |       |'
@ 24,2 SAY ' Next '
@ 24,9 SAY ' Prev '
@ 24,16 SAY ' First '
@ 24,24 SAY ' Last '
@ 24,38 SAY ' Search '
@ 24,47 SAY ' Edit '
@ 24,54 SAY ' Append '
@ 24,63 SAY ' Delete '
@ 24,72 SAY ' Quit '

DO WHILE .t.
help_code = 'inspect'

   * display entry record
   DO inspdisp


*************************************************************
***
   * user selects action here

*************************************************************
***
   SET COLOR TO &c_unselected

   @ 24,2 PROMPT ' Next '
   @ 24,9 PROMPT ' Prev '
   @ 24,16 PROMPT ' First '
   @ 24,24 PROMPT ' Last '
   @ 24,38 PROMPT ' Search '
   @ 24,47 PROMPT ' Edit '
   @ 24,54 PROMPT ' Append '
   @ 24,63 PROMPT ' Delete '
   @ 24,72 PROMPT ' Quit '

   MENU TO key

   * perform selected option
   DO CASE
      CASE key =1
           DO inspnext
      CASE key =2
           DO inspprev
      CASE key =3
           DO inspfirst
      CASE key =4
           DO insplast
      CASE key =5
           DO inspsearch
      CASE key =6
```

```
            DO inspedit
        CASE key =7
            DO inspapprec
        CASE key =8
            DO inspdelrec
        CASE key =9
            DO inspquit
            RETURN

    ENDCASE

ENDDO
RETURN
***************************** END OF MENU PROCEDURE
***********************

*********************************************************************
**************
*
* action procedures:
*
* the following are called by the menu loop above.
*
*********************************************************************
**************


***********************************
* inspnext: go to next record
***********************************
PROC inspnext
SKIP
IF eof()
   GOTO BOTTOM
   DO statmsg with 'End of file!'
ELSE
   DO statmsg with ''
   mLastOrg = ORG_NUMBER
   DO inspdisp
ENDIF
RETURN


***********************************
* inspprev: go to previous record
***********************************
PROC inspprev
SKIP -1
IF bof()
   GOTO TOP
   DO statmsg with 'Beginning of file!'
ELSE
   DO statmsg with ''
   mLastOrg = ORG_NUMBER
```

```
   DO inspdisp
ENDIF
RETURN


**********************************
* inspfirst: go to first record
**********************************
PROC inspfirst
GOTO TOP
DO statmsg with ''
   mLastOrg = ORG_NUMBER
DO inspdisp
RETURN


**********************************
* insplast: go to last record
**********************************
PROC insplast
GOTO BOTTOM
DO statmsg with ''
   mLastOrg = ORG_NUMBER
DO inspdisp
RETURN



*****************************************************************
*****************
* inspsearch: search for a record, given entry into desired
criterion fields
*****************************************************************
*****************
PROC inspsearch
PRIV searchval,orecno
help_code = 'inspsearch'
searchval = space(6)
DO statmsg with ''
@ 23,2 SAY 'Enter Org/Shop number: ' GET searchval PICTURE
'@! NNNNN'
READ
orecno = recno()
SEEK searchval
IF .NOT. FOUND()
   GOTO orecno
   DO statmsg with 'Organization number not found!'
ELSE
   DO statmsg with ''
   mLastOrg = ORG_NUMBER
   DO inspdisp
ENDIF
RETURN
```

```
**********************************
* inspedit: edit current record
**********************************
PROC inspedit

*** create memvar duplicates for all fields
PRIVATE mLAST_INSP, mNEXT_INSP, mDISC_DATE, mORG
PRIVATE mFOLLOW, mFOLLOW_UP, mSTATUS
help_code = 'inspedit'

*** ...and initialize 'em
mLAST_INSP = LAST_INSP
mNEXT_INSP = NEXT_INSP
mORG = ORG_NUMBER
mDISC_DATE = DISC_DATE
mFOLLOW = 'N'
mFOLLOW_UP = FOLLOW_UP
mSTATUS = STATUS

DO statmsg with 'Edit record. ^W to save; Esc to abandon'

*** get input fields into memvar duplicates
@ 3, 57 GET mLAST_INSP
@ 4, 61 GET mNEXT_INSP
@ 6, 50 GET mORG PICTURE '@! NNNNNN'
@ 9, 8 GET mDISC_DATE
READ
DO Fred
@ 13, 19 GET mFOLLOW PICTURE 'Y'
READ
IF mFOLLOW = 'Y'       && reinspection needed
   mFOLLOW_UP = mFOLLOW_UP + 30
   @ 13, 40 GET mFOLLOW_UP
ENDIF
@ 14, 10 GET mSTATUS PICTURE '@! AAAAA'
READ

IF lastkey() = 27       && user escaped out of READ
   RETURN
ENDIF

REPLACE LAST_INSP WITH mLAST_INSP
REPLACE NEXT_INSP WITH mNEXT_INSP
REPLACE ORG_NUMBER WITH UPPER(mORG)
REPLACE DISC_DATE WITH mDISC_DATE
REPLACE FOLLOW_UP WITH mFOLLOW_UP
REPLACE STATUS WITH UPPER(mSTATUS)

   mLastOrg = ORG_NUMBER
DO inspdisp
RETURN
```

```
**********************************
* inspapprec: append new record
**********************************
PROC inspapprec

*** create memvar duplicates for all fields
PRIVATE mLAST_INSP, mNEXT_INSP, mDISC_DATE, mDISCREPNCY,
mREFERENCE, mORG
PRIVATE mRESPONSE, mFOLLOW, mSTATUS
help_code = 'inspappnd'
*** ...and initialize 'em
mLAST_INSP = DATE()
mNEXT_INSP = DATE()+30
mORG = SPACE(5)
mDISC_DATE = DATE()
mFOLLOW_UP = DATE()
mFOLLOW = 'Y'
mSTATUS = 'OPEN'

*** Erase several blocks of the new record
@ 3, 9 SAY SPACE(10)
@ 4,13 SAY SPACE(12)
@ 5,11 SAY SPACE(25)
@ 6,18 SAY SPACE(10)
@ 3,57 SAY SPACE(8)
@ 4,61 SAY SPACE(8)
@ 9,8 SAY SPACE(8)
@ 10,16 CLEAR TO 12,76
@ 14,10 SAY SPACE(6)

DO statmsg with 'Enter new record. ^W to save; Esc to
abandon'

*** get input fields into memvar duplicates
@ 6, 50 GET mORG PICTURE '@! NNNNNN'
@ 3, 57 GET mLAST_INSP
@ 4, 61 GET mNEXT_INSP
@ 9, 8 GET mDISC_DATE
@ 13, 19 GET mFOLLOW PICTURE 'Y'
READ
IF mFOLLOW = 'Y'        && reinspection needed
   mFOLLOW_UP = mFOLLOW_UP + 30
   @ 13, 40 GET mFOLLOW_UP
   READ
ENDIF
@ 14, 10 GET mSTATUS PICTURE '@! AAAAAA'
READ

IF lastkey() = 27        && user escaped out of READ
   RETURN
ENDIF
```

```
* append into DBF fields
APPEND BLANK
REPLACE LAST_INSP WITH mLAST_INSP
REPLACE NEXT_INSP WITH mNEXT_INSP
REPLACE ORG_NUMBER WITH UPPER(mORG)
REPLACE DISC_DATE WITH mDISC_DATE
REPLACE FOLLOW_UP WITH mFOLLOW_UP
REPLACE STATUS WITH UPPER(mSTATUS)
DO Fred     && This way the memo fields are empty!
mBACKUP = .T.

REINDEX

   mLastOrg = ORG_NUMBER
DO inspdisp
RETURN

*************************************************************
********
* Fred : edit the DISCREPNCY, REFERENCE and RESPONSE Memo
fields
*************************************************************
********
PROC Fred

SET CURSOR ON
SET COLOR TO &c_selected
@ 16, 1, 21, 61 BOX '┌─┐│┘─└│'
help_code = 'fred'

DO statmsg with 'Enter Discrepancy info: ^W to Save, ESC to
abort'
REPLACE DISCREPNCY WITH MEMOEDIT(DISCREPNCY, 17,2,20,60,.T.)

DO statmsg with 'Enter Reference. ^W to Save, ESC to abort'
REPLACE REFERENCE  WITH MEMOEDIT(REFERENCE,  17,2,20,60,.T.)

DO statmsg with 'Enter User Response. ^W to Save, ESC to
abort'
REPLACE RESPONSE   WITH MEMOEDIT(RESPONSE,   17,2,20,60,.T.)

DO statmsg with ''
SET COLOR TO &c_unselected
SET CURSOR OFF
RETURN



***********************************************
* inspdelrec: delete record. If deleted, recall
***********************************************
PROC inspdelrec
IF .not. deleted()
   DELETE
```

```
   mINSPECT = .T.
ELSE
   RECALL
ENDIF
DO statmsg with ''
   mLastOrg = ORG_NUMBER
DO inspdisp
RETURN


**********************************
* inspquit: various quit things
**********************************
PROC inspquit
SET COLOR TO &c_normal
SET CURSOR OFF
CLEAR
RETURN


*****************************************************************
*********
* inspdisp: displays current record, along with recno() and
deleted()
*****************************************************************
*********
PROC inspdisp
PRIV searchval,mblank
SELECT 4
mLastArea = 4
@ 6, 50 GET ORG_NUMBER
searchval = ORG_NUMBER
SELECT 2                  && Switch to the ORGDATA database
mLastArea = 2
SEEK (searchval)
IF eof()
   DO statmsg with 'You need to define this organization.'
ELSE
   @ 3, 9  GET WING
   @ 4, 13 GET SQUADRON
   @ 5, 11 GET BRANCH
   @ 6, 18 GET OFFICE
ENDIF
SELECT 4                  && Switch back to the INSPECT
database
mLastArea = 4
@ 3, 57 GET LAST_INSP
@ 4, 61 GET NEXT_INSP
@ 9, 8 GET DISC_DATE
mblank = space(60)
@ 10, 16 CLEAR TO 12, 76
@ 10, 16 SAY MEMOLINE(DISCREPNCY,60,1)
@ 11, 16 SAY MEMOLINE(REFERENCE,60,1)
@ 12, 16 SAY MEMOLINE(RESPONSE,60,1)
```

```
@ 17, 2 CLEAR TO 20, 60
@ 13, 40 GET FOLLOW_UP
@ 14, 10 GET STATUS
CLEAR GETS
@ 23,70 SAY iif(deleted(),'Deleted','       ')

SET COLOR TO &c_unselected
RETURN
***
*** C:\CLIP\LISTINGS.PRG : Main menu.
*** Generated March 17, 1990 from C:\UI\WW\LISTINGS.WW
*** Target environment: Clipper Summer 87
***

* menu initialization
PRIVATE optkeys, numopts, oldchoice, newchoice, key, mTemp,
mTime, mDate
optkeys = "ADIOQ"
numopts = 5
oldchoice = 0
newchoice = 1
key =0
mKey = 78
help_code = 'listings'

* inkey aliases
PRIVATE RKup,RKdn,RKrt,RKlt,RKret,RKspc
RKup = 5
RKdn = 24
RKrt = 4
RKlt = 19
RKret = 13
RKspc = 32

* main loop: iterates once for each time an option action is
performed
DO WHILE .t.

  * display the menu screen
  CLEAR
  SET COLOR TO &c_normal
  @ 0, 0, 4, 80 BOX "┌┐│┘─└│ "
  @ 2, 27 SAY "Reports and Listings Menu"

  SET COLOR TO &c_normal
  @ 6, 0, 9, 40 BOX "┌┐│┘─└│ "
  SET COLOR TO &c_unselected
  @ 7, 2 SAY "A  Report of all Authorized"
  @ 8, 2 SAY "Items  (All WRM Auth's)"

  SET COLOR TO &c_normal
  @ 10, 0, 13, 40 BOX "┌┐│┘─└│ "
  SET COLOR TO &c_unselected
  @ 11, 2 SAY "D  Report of all Authorized"
```

```
  @ 12, 2 SAY "Items and all details"

  SET COLOR TO &c_normal
  @ 14, 0, 17, 40 BOX "┌┐│┘─└│ "
  SET COLOR TO &c_unselected
  @ 15, 2 SAY "I  Report of all Inspections"

  SET COLOR TO &c_normal
  @ 18, 0, 21, 40 BOX "┌┐│┘─└│ "
  SET COLOR TO &c_unselected
  @ 19, 2 SAY "O  Report of all Organizations"
  @ 20, 2 SAY "Points of Contact info"

  SET COLOR TO &c_normal
  @ 22, 0, 24, 40 BOX "┌┐│┘─└│ "
  SET COLOR TO &c_unselected
  @ 23, 2 SAY "Q  Return to Main Menu"

  * keyhit loop: iterates ones for each key input, breaks on
selection
  DO WHILE .t.

    * if selected option has changed, update the bounce-bar
    IF oldchoice<>newchoice
      * highlight new option
      DO CASE

     CASE newchoice =1
          SET COLOR TO &c_normal
          @ 6, 0, 9, 40 BOX "┌┐│┘─└│"
          SET COLOR TO &c_selected
          @ 7,1,8,39 BOX "          "
          @ 7, 2 SAY "A  Report of all Authorized"
          @ 8, 2 SAY "Items  (All WRM Auth's)"
               help_code = 'allauth'

     CASE newchoice =2
          SET COLOR TO &c_normal
          @ 10, 0, 13, 40 BOX "┌┐│┘─└│"
          SET COLOR TO &c_selected
          @ 11,1,12,39 BOX "          "
          @ 11, 2 SAY "D  Report of all Authorized"
          @ 12, 2 SAY "Items and all details"
               help_code = 'authplus'

     CASE newchoice =3
          SET COLOR TO &c_normal
          @ 14, 0, 17, 40 BOX "┌┐│┘─└│"
          SET COLOR TO &c_selected
          @ 15,1,16,39 BOX "          "
          @ 15, 2 SAY "I  Report of all Inspections"
               help_code = 'dueinsp'

     CASE newchoice =4
```

```
      SET COLOR TO &c_normal
      @ 18, 0, 21, 40 BOX "┌─┐│┘─└│"
      SET COLOR TO &c_selected
      @ 19,1,20,39 BOX "          "
      @ 19, 2 SAY "O  Report of all Organization"
      @ 20, 2 SAY "Points of Contact info"
           help_code = 'pocinfo'

 CASE newchoice =5
      SET COLOR TO &c_normal
      @ 22, 0, 24, 40 BOX "┌─┐│┘─└│"
      SET COLOR TO &c_selected
      @ 23,1,23,39 BOX "          "
      @ 23, 2 SAY "Q  Return to Main Menu"
           help_code = 'listings'

  ENDCASE

  * reset oldchoice for another pass
  oldchoice =newchoice
ENDIF

* if return has been hit
IF key=RKret .or. at(upper(chr(key)), optkeys) > 0
  SET COLOR TO &c_normal
  * fall out to action loop
  EXIT
ENDIF

* get key input
key=inkey(0)

* update choice number based on key response
DO CASE
  * down/right arrow: increment choice or wrap
  CASE key =RKdn .or. key =RKrt .or. key = RKspc
 newchoice = iif(oldchoice=numopts,1,oldchoice+1)
  * up/left arrow: decrement choice or wrap
  CASE key = RKup .or. key = RKlt
 newchoice = iif(oldchoice=1,numopts,oldchoice-1)
  * option trigger: set choice to option
  CASE at(upper(chr(key)), optkeys) > 0
 newchoice = at(upper(chr(key)), optkeys)
ENDCASE

IF oldchoice<>newchoice
  * lowlight old option
  DO CASE

 CASE oldchoice =1
      SET COLOR TO &c_normal
      @ 6, 0, 9, 40 BOX "┌─┐│┘─└│"
      SET COLOR TO &c_unselected
      @ 7,1,8,39 BOX "          "
```

```
          @ 7, 2 SAY "A  Report of all Authorized"
          @ 8, 2 SAY "Items  (All WRM Auth's)"

      CASE oldchoice =2
          SET COLOR TO &c_normal
          @ 10, 0, 13, 40 BOX "┌─┐│┘─└│"
          SET COLOR TO &c_unselected
          @ 11,1,12,39 BOX "         "
          @ 11, 2 SAY "D  Report of all Authorized"
          @ 12, 2 SAY "Items and all details"
      CASE oldchoice =3
          SET COLOR TO &c_normal
          @ 14, 0, 17, 40 BOX "┌─┐│┘─└│"
          SET COLOR TO &c_unselected
          @ 15,1,16,39 BOX "         "
          @ 15, 2 SAY "I  Report of all Inspections"

      CASE oldchoice =4
          SET COLOR TO &c_normal
          @ 18, 0, 21, 40 BOX "┌─┐│┘─└│"
          SET COLOR TO &c_unselected
          @ 19,1,20,39 BOX "         "
          @ 19, 2 SAY "O  Report of all Organization"
          @ 20, 2 SAY "Points of Contact info"

      CASE oldchoice =5
          SET COLOR TO &c_normal
          @ 22, 0, 24, 40 BOX "┌─┐│┘─└│"
          SET COLOR TO &c_unselected
          @ 23,1,23,39 BOX "         "
          @ 23, 2 SAY "Q  Return to Main Menu"


        ENDCASE
      ENDIF

    * end of keyhit loop
    ENDDO

IF newchoice <> 5
    CLEAR GETS
    mTemp = 'N'
    SET COLOR TO &c_selected
    @ 10, 42, 14, 70 BOX "┌─┐│┘─└│ "
    @ 11, 44 SAY "Send Report to Printer?"
    @ 12, 55 SAY 'Y/N' GET mTemp PICTURE "Y"
    READ
ENDIF

    * perform selected option
    DO CASE

      CASE newchoice =1
        SELECT 1
```

```
        mLastArea = 1
        IF mTemp = 'Y'
          REPORT FORM All_Auth TO PRINT
        ELSE
          REPORT FORM All_Auth ALL
        ENDIF
        RETURN

      CASE newchoice =2
        SELECT 4
        mLastArea = 4
         DO AllAuth2 with mTemp
        RETURN

      CASE newchoice =3
        SELECT 4
        mLastArea = 4
         CLEAR GETS
         mTime = 30
         @ 10, 42, 14, 75 BOX '┌┐│┘─└│ '
         @ 11, 44 SAY 'How many days from today?' GET mTime
PICTURE '999'
         READ
         mDate = DATE() + mTime
        IF mTemp = 'Y'
          REPORT FORM INSPECT FOR NEXT_INSP <= mDate TO PRINT
        ELSE
          REPORT FORM INSPECT FOR NEXT_INSP <= mDate
        ENDIF
        RETURN

      CASE newchoice =4
        SELECT 2
        mLastArea = 2
        IF mTemp = 'Y'
          REPORT FORM AllOrgs TO PRINT
        ELSE
          REPORT FORM AllOrgs
        ENDIF
        RETURN

      CASE newchoice =5
        RETURN

    ENDCASE

    * set old choice var to 0 so we get a highlight on the
current option
    oldchoice = 0

    * and set key input var to 0 so we don't fall out again
    key = 0

ENDDO
```

```
RETURN***
*** C:\CLIP\READIN.PRG : Main menu.
*** Generated March 18, 1990 from
*** Target environment: Clipper Summer 87
***

* environment
SET SCOREBOARD off
SET CONFIRM off

* menu initialization
PRIVATE optkeys, numopts, oldchoice, newchoice, key
optkeys = 'RBDQ'
numopts = 4
oldchoice = 0
newchoice = 1
key =0
help_code = 'readin'

* inkey aliases
PRIVATE RKup,RKdn,RKrt,RKlt,RKret,RKspc
RKup = 5
RKdn = 24
RKrt = 4
RKlt = 19
RKret = 13
RKspc = 32

* main loop: iterates once for each time an option action is
performed
DO WHILE .t.

   * display the menu screen
   SET COLOR TO &c_normal
   CLEAR
   SET COLOR TO &c_normal
   @ 0, 0, 3, 79 BOX '┌┐│┘─└│'
   SET COLOR TO &c_selected
   @ 1,1,2,78 BOX '         '
   @ 1, 28 SAY 'DataBase Management Menu'
   SET COLOR TO &c_unselected
   @ 4, 0, 8, 31 BOX '┌┐│┘─└│ '
   @ 5, 1 SAY 'R Read In SBSS Data'
   @ 6, 3 SAY '(Replaces all authorization'
   @ 7, 2 SAY 'and supply detail documents)'
   @ 10, 0, 14, 31 BOX '┌┐│┘─└│ '
   @ 11, 1 SAY 'B   Backup current DataBases'
   @ 12, 5 SAY '(Do this weekly!)'
   @ 16, 0, 20, 31 BOX '┌┐│┘─└│ '
   @ 17, 1 SAY 'D  Restore previously saved'
   @ 18, 5 SAY 'DataBases (ie, the ones'
   @ 19, 4 SAY 'you backed up weekly...)'
   @ 10, 40, 14, 70 BOX '┌┐│┘─└│ '
   @ 12, 42 SAY 'Q  Return to Main Menu'
```

```
   * keyhit loop: iterates ones for each key input, breaks
on selection
   DO WHILE .t.

      * if selected option has changed, update the
bounce-bar
      IF oldchoice<>newchoice
         * highlight new option
         DO CASE

            CASE newchoice =1
                 SET COLOR TO &c_normal
                 @ 4, 0, 8, 31 BOX '┌┐│┘─└│'
                 SET COLOR TO &c_selected
                 @ 5,1,7,30 BOX '          '
                 @ 5, 1 SAY 'R Read In SBSS Extracted Data'
                 @ 6, 3 SAY '(Replaces all authorization'
                 @ 7, 2 SAY 'and supply detail documents)'
                 help_code = 'extract'

            CASE newchoice =2
                 SET COLOR TO &c_normal
                 @ 10, 0, 14, 31 BOX '┌┐│┘─└│'
                 SET COLOR TO &c_selected
                 @ 11,1,13,30 BOX '          '
                 @ 11, 1 SAY 'B   Backup current DataBases'
                 @ 12, 5 SAY '(Do this weekly!)'
                 help_code = 'backup'

            CASE newchoice =3
                 SET COLOR TO &c_normal
                 @ 16, 0, 20, 31 BOX '┌┐│┘─└│'
                 SET COLOR TO &c_selected
                 @ 17,1,19,30 BOX '          '
                 @ 17, 1 SAY 'D  Restore previously saved'
                 @ 18, 5 SAY 'DataBases (ie, the ones'
                 @ 19, 4 SAY 'you backed up weekly...)'
                 help_code = 'restore'

            CASE newchoice =4
                 SET COLOR TO &c_normal
                 @ 10, 40, 14, 70 BOX '┌┐│┘─└│'
                 SET COLOR TO &c_selected
                 @ 11,41,13,69 BOX '          '
                 @ 12, 42 SAY 'Q  Return to Main Menu'
                 help_code = 'readin'

         ENDCASE

         * reset oldchoice for another pass
         oldchoice =newchoice
      ENDIF
```

```
      * if return or an option trigger has been hit, perform
the option
      IF key=RKret .or. at(upper(chr(key)), optkeys) >0
         SET COLOR TO &c_normal
         * fall out to action loop
         EXIT
      ENDIF

      * get key input
      key=inkey(0)

      * update choice number based on key response
      DO CASE
            * down/right arrow: increment choice or wrap
         CASE key =RKdn .or. key =RKrt .or. key = RKspc
            newchoice =
iif(oldchoice=numopts,1,oldchoice+1)
            * up/left arrow: decrement choice or wrap
         CASE key = RKup .or. key = RKlt
            newchoice =
iif(oldchoice=1,numopts,oldchoice-1)
            * option trigger: set choice to option
         CASE at(upper(chr(key)), optkeys) > 0
            newchoice = at(upper(chr(key)), optkeys)
      ENDCASE

      IF oldchoice<>newchoice
         * lowlight old option
         DO CASE

            CASE oldchoice =1
                 SET COLOR TO &c_normal
                 @ 4, 0, 8, 31 BOX '┌┐│┘─└│'
                 SET COLOR TO &c_unselected
                 @ 5,1,7,30 BOX '         '
                 @ 5, 1 SAY 'R Read In SBSS Extracted Data'
                 @ 6, 3 SAY '(Replaces all authorization'
                 @ 7, 2 SAY 'and supply detail documents)'

            CASE oldchoice =2
                 SET COLOR TO &c_normal
                 @ 10, 0, 14, 31 BOX '┌┐│┘─└│'
                 SET COLOR TO &c_unselected
                 @ 11,1,13,30 BOX '         '
                 @ 11, 1 SAY 'B   Backup current DataBases'
                 @ 12, 5 SAY '(Do this weekly!)'

            CASE oldchoice =3
                 SET COLOR TO &c_normal
                 @ 16, 0, 20, 31 BOX '┌┐│┘─└│'
                 SET COLOR TO &c_unselected
                 @ 17,1,19,30 BOX '         '
                 @ 17, 1 SAY 'D  Restore previously saved'
                 @ 18, 5 SAY 'DataBases (ie, the ones'
```

```
                    @ 19, 4 SAY 'you backed up weekly...)'

               CASE oldchoice =4
                    SET COLOR TO &c_normal
                    @ 10, 40, 14, 70 BOX '┌┐│┘─└│'
                    SET COLOR TO &c_unselected
                    @ 11,41,13,69 BOX '         '
                    @ 12, 42 SAY 'Q  Return to Main Menu'

            ENDCASE
         ENDIF

         * end of keyhit loop
      ENDDO


      * perform selected option
      DO CASE

         CASE newchoice =1
              DO NewDataIn
              RETURN

         CASE newchoice =2
              DO Backup
              RETURN

         CASE newchoice =3
              DO Restore
              RETURN

         CASE newchoice =4
              RETURN

      ENDCASE

      * set old choice var to 0 so we get a highlight on the
current option
      oldchoice =0

      * and set key input var to 0 so we don't fall out again
      key =0

ENDDO
RETURN

*********************************************************
* Replace existing Basic and Document DataBases with
* SBSS output using a QLP or SURGE program
*********************************************************
PROC NewDataIn
PRIVATE mTemp,orecno

mTemp = 'N'
```

```
mAvail = .F.
CLEAR GETS
* Need to get a confirmation here before executing
SET COLOR TO &c_selected
@ 19, 15, 21, 55 BOX '┌┐│┘─└│ '
@ 19, 19 SAY 'Are you sure you want to do this?'
@ 20, 17 GET mTemp PICTURE 'Y'
READ
IF mTemp = 'Y'
   * Open a window to enter drive path designation
   DO GetDrive
   IF mAvail = .T.
      SELECT 9
      mLastArea = 9
      USE TEMP
      APPEND FROM A:\WRM.DAT WHILE .NOT. eof() SDF

      SELECT 1
      mLastArea = 1
      ZAP            && clean out document.dbf

      SELECT 3
      mLastArea = 3
      ZAP            && clean out basic.dbf

      SELECT 9       && Use Temp.dbf
      mLastArea = 9
      GOTO TOP
       orecno = RECNO()

      DO WHILE .NOT. eof()       && Begin While Loop
         GOTO orecno
         mTemp = SUBSTR(TEMP->Big,1,3)
         IF mTemp ='EOF'
            SKIP
            orecno = RECNO()
         ELSE
            mTemp = SUBSTR(TEMP->Big,94,1)
            IF mTemp = 'O'             && Due-Out
               * Extract into DOCUMENT.DBF
               SELECT 3
               mLastArea = 3
               * extract data elements into a new record
               APPEND BLANK
               REPLACE STOCK_NBR WITH SUBSTR(TEMP->Big,1,15)
               REPLACE DOC_NUMBER WITH
SUBSTR(TEMP->Big,16,14)
               REPLACE DOC_TYPE WITH 'O'
               REPLACE QTY_ORDERD WITH
VAL(SUBSTR(TEMP->Big,30,5))
       REPLACE DUE_IN_DOC WITH
'I'+SUBSTR(TEMP->Big,17,5)+SUBSTR(TEMP->Big,52,8)
       * Creates a Due_In_Document number
               REPLACE STATUS WITH SUBSTR(TEMP->Big,60,2)
```

```
                REPLACE EDD WITH VAL(SUBSTR(TEMP->Big,62,4))
             ELSE
                * Other details are MSK                - 'Q'
                * WRM/WCDO authorizations        - 'W'
                * High Priority MSK                    - 'U'
                * Airborne and Non-Airborne WRSK - 'N'
                * Extract into BASIC.DBF
                SELECT 1
                mLastArea = 1
                * extract data elements into a new record
                APPEND BLANK
                REPLACE STOCK_NBR WITH SUBSTR(TEMP->Big,1,15)
                REPLACE ATHRZD_QTY WITH
VAL(SUBSTR(TEMP->Big,30,5))
                REPLACE QTY_UNSUPP WITH
VAL(SUBSTR(TEMP->Big,43,5))
                mNET_QTY = ATHRZD_QTY - QTY_UNSUPP
                mQTY_SHORT = 0
                REPLACE NET_QTY WITH mNET_QTY
                REPLACE QTY_ONHAND WITH mNET_QTY
                REPLACE QTY_SHORT WITH mQTY_SHORT
                REPLACE SOURCE_DOC WITH
SUBSTR(TEMP->Big,16,14)
                REPLACE DOC_TYPE WITH SUBSTR(TEMP->Big,94,1)
                REPLACE NOMENCLATR WITH
SUBSTR(TEMP->Big,52,19)
                REPLACE UNIT_PRICE WITH
VAL(SUBSTR(TEMP->Big,71,10))/100
                REPLACE ERRC_CODE WITH SUBSTR(TEMP->Big,82,3)
                REPLACE UNIT_ISSUE WITH
SUBSTR(TEMP->Big,85,2)
                REPLACE ISG_NUMBER WITH
SUBSTR(TEMP->Big,87,4)
                REPLACE ISG_RELAT WITH SUBSTR(TEMP->Big,91,1)
                REPLACE SHELF_LIFE WITH
SUBSTR(TEMP->Big,92,1)
             ENDIF
             SELECT 9   && Use Temp.dbf
             mLastArea = 9
             SKIP       && Goto next record of temp.dbf
             orecno = RECNO()
          ENDIF
       ENDDO          && End of While loop

       SELECT 1       && Reconstruct index files
       mLastArea = 1
       REINDEX

       SELECT 3
       mLastArea = 3
       REINDEX

       SELECT 9
       mLastArea = 9
```

```
      ZAP                 && CLEAN OUT TEMP.DBF

      DO Checking         && Checks for new NSN or Org/Shop
codes
   ENDIF
ELSE
   RETURN
ENDIF

RETURN


*****************************************************************
****
* Copy existing DataBases and Index Files to A:\ drive
*****************************************************************
****
PROC Backup
* Open a window to enter drive path designation
DO GetDrive
IF mAvail = .T.
   CLEAR SCREEN
   * copy database files to destination disk
   RUN COPY *.DBF A:\*.DBF
   CLEAR SCREEN
   * copy memo files to destination disk
   RUN COPY *.DBT A:\*.DBT
   CLEAR SCREEN
   * copy index files to destination disk
   RUN COPY *.NTX A:\*.NTX
ENDIF
RETURN

*************************************************************
* Copy files from A:\ drive to working directory
*************************************************************
PROC Restore
* Open a window to enter source drive path
DO GetDrive
IF mAvail = .T.
   CLEAR SCREEN
   * copy databases from backup disk
   RUN COPY A:\*.DBF *.DBF
   CLEAR SCREEN
   * copy memo files from backup disk
   RUN COPY A:\*.DBT *.DBT
   CLEAR SCREEN
   * copy index files from backup disk
   RUN COPY A:\*.NTX *.NTX
ENDIF
RETURN


*********************************************************
```

```
* GetDrive checks for drive availability, and sets
* mAvail to .T. or .F.
*********************************************************
PROC GetDrive
PUBLIC mAvail
PRIVATE mTemp

mAvail = FCREATE('A:\Temp.txt')
IF FERROR() <> 0
     mAvail = .F.
     * Print an error message box
     @ 19, 15, 21, 55 BOX '┌─┐│┘─└│ '
     @ 20, 19 SAY 'Cannot access the A: drive...'
     DO Statmsg with 'Press any key to continue.'
     mTemp = INKEY(0)
ELSE
   mAvail = .T.
   DO statmsg with ''
ENDIF
RETURN
*****************************************************************
************
* PROC CHECKING - BY Kevin Tanzer
* - Check the Document.dbf database to ensure that a NSN
listed in the
* dbf exists in the Basic.dbf, if not it will add that NSN
with an
* UNKNOWN stock number.
* - Will also check that any organization listed
* in the Document.dbf exists in the Organiz.dbf, if not it
will add that
* organization number.
*****************************************************************
************
PRIV orecno,searchval,searchval2

DO Statmsg with 'Checking for new NSNs'

SELECT 3
GOTO TOP
orecno = RECNO()

DO WHILE .NOT. EOF()
  searchval = STOCK_NBR
  searchval2 = SUBSTR(DOC_NUMBER,2,5)

  SELECT 1
  SEEK searchval
  IF .NOT. FOUND()
    APPEND BLANK
    REPLACE STOCK_NBR WITH searchval
    REPLACE NOMENCLATR WITH 'UNKNOWN'
  ENDIF
```

```
  SELECT 2
  LOCATE FOR ORG_NUMBR = searchval2
  IF .NOT. FOUND()
    APPEND BLANK
    REPLACE ORG_NUMBR WITH searchval2
    REPLACE OFFICE WITH 'UNKNOWN'
  ENDIF

  SELECT 3
  GOTO orecno
  SKIP
  orecno = RECNO()
ENDDO

SELECT 1
REINDEX

SELECT 2
  SET UNIQUE ON        && Prevents duplicate org/shop codes
  REINDEX
  SET UNIQUE OFF
  mLastArea = 2

RETURN
***
*** ClassPop created on 2 Apr by Kevin Tanzer
***
***

PRIVATE mKey
mKey = 89

SET COLOR TO &c_flashing
@ 10,5,15,74 BOX "┌┐└┘─│ "
@ 12,20 SAY "DO NOT USE FOR CLASSIFIED INFORMATION!!!"
SET COLOR TO &c_normal
DO statmsg WITH "Press any key to continue..."
mKey = INKEY(0)
CLEAR SCREEN

RETURN
***
***  ClassScr.prg created 2 Apr 90 by Kevin Tanzer
***
***

PRIVATE mKey
mKey = 89

CLEAR SCREEN
CLEAR GETS
SET COLOR TO &c_normal
     @ 0,0,23,79 BOX "┌┐└┘─│ "
     @ 3,29 SAY "SIMBL V2.0 (PROTOTYPE)"
```

```
     @ 5,17 SAY "THIS PROGRAM IS NOT CLEARED FOR CLASSIFIED
DATA"
     SET COLOR TO &c_flashing
     @ 10,5,15,74 BOX "┌┐│┘─└│ "
     @ 12,20 SAY "DO NOT USE FOR CLASSIFIED INFORMATION!!!"
     SET COLOR TO &c_normal
    DO statmsg with "Press any key to continue..."
     mKey = INKEY(0)
CLEAR SCREEN
RETURN
*** CLEANUP PROCEDURE
*** AUTHOR:  CPT KEVIN TANZER ON 16 MAR 90
*** LAST UPDATE ON 21 APR 90
*** PACK DATABASES AND REINDEX

PRIVATE mKey
mKey = 'N'

IF mBACKUP = .T.
  SET COLOR TO &c_normal
  @ 10,5,16,74 BOX "┌┐│┘─└│ "
  CLEAR GETS
  @ 12,7 SAY "You have at added some data to the program,
would"
  @ 14,10 SAY "you like to back up the new data?" GET mKey
PICTURE "Y"
  READ
  IF mKey = 'Y'
    DO Backup
    RETURN
  ENDIF
ENDIF

IF mBASIC = .T.
  SET COLOR TO &c_normal
  @ 10,5,16,74 BOX "┌┐│┘─└│ "
  CLEAR GETS
 @ 12,7 SAY "You have at least one authorization record
marked for"
 @ 14,10 SAY "deletion, are you sure you want to do this?"
GET mKey PICTURE "Y"
  SET COLOR TO &c_flashing
  DO statmsg with "Press Y to confirm deletion, any other
key to abort"
  READ
  IF mKey = "Y"
     SELECT 1
     mLastArea =1
     PACK
     REINDEX
  ELSE
    SELECT 1
    mLastArea = 1
    RECALL ALL
```

```
    DO statmsg with "Recovered all authorization records"
     mKey = INKEY(5)
  ENDIF
ENDIF

IF mORGDATA = .T.
  SET COLOR TO &c_normal
  @ 10,5,16,74 BOX "┌─┐│┘─└│ "
  @ 12,7 SAY "You have at least one organizational record
marked for"
 @ 14,10 SAY "deletion, are you sure you want to do this?"
GET mKey PICTURE "Y"
  SET COLOR TO &c_flashing
  DO statmsg with "Press Y to confirm deletion, any other
key to abort"
  READ
  IF mKey = "Y"
     SELECT 2
     mLastArea =2
    PACK
    REINDEX
  ELSE
    SELECT 2
    mLastArea = 2
    RECALL ALL
    DO statmsg with "Recovered all organizational records"
    mKey = INKEY(5)
  ENDIF
ENDIF

IF mDOCUMENT = .T.
  SET COLOR TO &c_normal
  @ 10,5,16,74 BOX "┌─┐│┘─└│ "
  @ 12,7 SAY "You have at least one document record marked
for"
 @ 14,10 SAY "deletion, are you sure you want to do this?"
GET mKey PICTURE "Y"
  SET COLOR TO &c_flashing
  DO statmsg with "Press Y to confirm deletion, any other
key to abort"
  READ
  IF mKey = "Y"
    SELECT 3
    mLastArea = 3
    PACK
    REINDEX
  ELSE
    SELECT 3
    mLastArea = 3
    RECALL ALL
    DO statmsg with "Recovered all document records"
    mKey = INKEY(5)
  ENDIF
ENDIF
```

```
IF mINSPECT = .T.
  SET COLOR TO &c_normal
  @ 10,5,16,74 BOX "┌─┐│┘─└│ "
  @ 12,7 SAY "You have at least one inspection record marked
for"
 @ 14,10 SAY "deletion, are you sure you want to do this?"
GET mKey PICTURE "Y"
  SET COLOR TO &c_flashing
  DO statmsg with "Press Y to confirm deletion, any other
key to abort"
  READ
  IF mKey = "Y"
    SELECT 4
    mLastArea = 4
    PACK
    REINDEX
  ELSE
    SELECT 4
    mLastArea = 4
    RECALL ALL
    DO statmsg with "Recovered all inspection records"
    mKey = INKEY(5)
  ENDIF
ENDIF

*** CLOSE DATABASES, INDEXES
CLOSE ALL
SET COLOR TO &c_normal
CLEAR SCREEN
SET CURSOR ON
QUIT************************************************************
*************
* PROC Doc_Look - Kevin Tanzer
* Translates a single parameter (1 letter) into a string,
* the letter is a detail type code, and returns a
plain-english
* description of the detail type
************************************************************
*********
PARAMETERS d_type
PRIV mtemp

mtemp = d_type
DO CASE
     CASE mtemp = 'O'
       mDOC = 'Due Out'

     CASE mtemp = 'W'
       mDOC = 'WRM/WCDO'

     CASE mtemp = 'N'
       mDOC = 'NAWRSK'
```

```
      CASE mtemp = 'U'
        mDOC = 'WRSK/HPMSK'

      CASE mtemp = 'Q'
        mDOC = 'MSK'

      CASE mtemp = 'I'
        mDOC = 'Due In'

ENDCASE
RETURN**********************************************************
*****************
* HELP.PRG - A context sensitive help program for use with
the
* Software for Inventory Management for the Base Logistician
(SIMBL)
*
* Written by Kevin M. Tanzer on 24 Apr 90
* Assumes a global variable HELP_CODE has been declared and
defined already.
* The three parameters are mandatory for the program, but
only
* the call_prg is used.  It identifies the subroutine that
called for Help.
* The other two parameters could be used for very
context-sensitive help, but
* requires a lot more work.
**************************************************************
***********
PARAMETERS call_prg, line_num, input_var

PRIV top_row,left_col,bottom_row,right_col
* variables for screen locations

IF call_prg = "HEL"    && Avoids recursive calling for Help
  RETURN
ENDIF

top_row = 5
left_col = 8
bottom_row = 18
right_col = 72

SELECT 9
USE Help

LOCATE FOR HELP_KEY = help_code       && Help.dbf is not
indexed

IF FOUND()
     SAVE SCREEN
     * Create a box on the screen, and use the MemoEdit
function to display the
```

```
     * contents of the memo field associated with the
HELP_KEY field.   This
     * is a variation of the help program from the Clipper
manual.

     @ top_row-2, left_col-2 CLEAR TO bottom_row-1,
right_col-1
     * Draws a shadow box underneath the Help box

     @ top_row-1, left_col-1, bottom_row+1, right_col+1 BOX
"┌─┐│┘─└│ "
     @ top_row-1,35 SAY " HELP BOX "

MEMOEDIT(HELP_TEXT,top_row,left_col,bottom_row,right_col,.F.
)
     * Displays the memo field contents without editing

     RESTORE SCREEN
ELSE
     DO statmsg with "No help available..."
ENDIF

* Selects the database area that was active prior to the
Help call,
* otherwise you cause a mismatch upon returning to the
calling routine.

IF mLastArea = 1
  SELECT 1
ENDIF

IF mLastArea = 2
  SELECT 2
ENDIF

IF mLastArea = 3
  SELECT 3
ENDIF

IF mLastArea = 4
  SELECT 4
ENDIF

RETURN
```

Appendix D: Research Participants

1. Brown, Captain Bryan K., HQ TAC, TAC/LGXW, Langley AFB VA.

2. Burke, Captain Mike, HQ PACAF, PACAF/LGXW, Hickam AFB HI.

3. Dacyk, Captain Peter, 52 TFW/LGXW, Spangdahlem AB Germany.

4. Degraffinreid, Technical Sergeant Willy L., HQ MAC/LGXW, Scott AFB IL.

5. Ellenburg, Mr. Charles, 313 AD/LGXW, Kadena AB Japan.

6. Graham, Captain William, 43 BMW/LGXW, Anderson AFB Guam.

7. Hagel, Major Stephen A., Air Force Logistics Management Center, LMC/LGXW, Gunter AFB AL.

8. Holck, Captain Brad C., HQ USAFE, USAFE/LGXW, Ramstein AB Germany.

9. MacDougal, Captain Damon L., 833 AD/LGXW, Holloman AFB NM.

10. Mathews, Sergeant Williams Jr., 363 TFW/LGXW, Shaw AFB SC.

11. Meyers, Mr. Thomas P., 3200 SPTW/LGX, Eglin AFB FL.

12. Michell, First Lieutenant Kimberly A., 92 BMW/LGXW, Fairchild AFB WA.

13. Payne, Mr. Edward, 3 SUPS/LGXW, Clark AB Phillipine.

14. Stock, Chief Master Sergeant Joeseph P., HQ SAC, HQ SAC/LGXW, Offut AFB NE.

15. Williams, Staff Sergeant Gerald R., 437 MAW/LGXW, Charleston AFB SC.

Bibliography

1. Atre, Shaku. Data Base: Structured Techniques for Design, Performance, and Management. New York: John Wiley & Sons, 1988.

2. Barry, TSgt William L., NCOIC Computer Support Section. Personal Interview. 4750th ABW Supply Squadron, Wright-Patterson AFB OH, 1 January 1989.

3. Beard, Maj Phillip H. A Database Management System Application for the Graduate Programs Office of the School of Systems and Logistics, Volume 1: Development and User's Manual. MS Thesis, AFIT/GLM/LSG/88S-3. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1988 (AD-A201557).

4. Bice, Lt Col Don L. Problems in the War Readiness Materiel Equipment Prepositioning Program. Air Command and Staff College, Maxwell AFB AL, June 1967 (AD-A953192).

5. Brown, Capt Bryan K., WRM Plans and Programs, HQ TAC/LGXW. Telephone Interview. Langely AFB VA, 6 November 1989.

6. Buck, Capt Stephen D. A DataBase Management System to Manipulate Data Collected at the National Training Center, Ft. Irwin, CA. MS Thesis, Naval Postgraduate School, Monterey CA, June 1987 (AD-A183197).

7. Burke, Capt Mike., WRM Plans and Programs, HQ PACOPS (Pacific Air Forces). Telephone Interview. Hickam AFB HI, 7 December 1989.

8. Dacyk, Capt Peter., Logistics Plans and Programs. Telephone Interview. Spangdahlem AB Germany, 7 December 1989.

9. Degraffinreid, TSgt Willy L., NCOIC WRM Plans and Programs HQ MAC/LGXW. Telephone Interview. Scott AFB IL, 7 December 1989.

10. Demers, W.A. "Too Many Spares?," Military Forum: 60-64 (March 1989).

11. Department of the Air Force. War Reserve Materiel (WRM) Policy. AFR 400- 24. Washington DC: HQ USAF, 28 November 1986.

12. Elbra, R. A. Database for the Small Computer User. Manchester England: NCC Publications, 1982.

13. Emory, William C. Business Research Methods. Homewood Illinois: Irwin, 1985.

14. Hagel, Maj Stephen A., Air Force Logistics Management Center, AFLMC/LGXW. Telephone Interview. Gunter AFB AL, 4 October through 14 November, 1989.

15. Holck, Capt Brad C., Plans and Programs USAFE/LGXW. Telephone Interview. Ramstein AB Germany, 7 December 1989.

16. House, William C. Interactive Decision Oriented Database Systems. New York: Petrocelli/Charter, 1977.

17. Kim, Maj Sam Nam and Capt Jae Bock Park. Application of a DataBase System for Korean Military Personnel Management. MS Thesis. Naval Postgraduate School, Monterey CA, March 1987 (AD-A181663).

18. Kroenke, David. Database Processing. Chicago: Science Research Associates, Inc., 1977.

19. Liskin, Miriam. "Which Dbase Is Right for You?," Personal Computing, 10: 113-121 (June 1986).

20. MacDougal, Capt Damon L., Chief, Logistics Plans and Programs. Telephone Interview. Holloman AFB NM, 22 June 1989.

21. Martin, James A. Principles of Database Management. Englewood Cliffs NJ: Prentice-Hall, Inc., 1976.

22. Mitchell, Lt Kimberly A., Chief, Logistics Plans and Programs. Telephone Interview. Fairchild AFB WA, 5 Jan 1990.

23. Peppers, Jerome G., History of United States Military Logistics: 1935-1985. Huntsville AL: Logistics Education Foundation Publishing, 1988.

24. Ritchhart, Maj Kenneth M. and Maj Robert L. Simmons, The Student Mix Software System (SMSS). Air Command and Staff College, Maxwell AFB AL, April 1986 (AD-A166689).

25. Smith, Lt Claire C. Supply Hotlist Report Generation for Fleet Ballistic Submarine Management Meetings. MS Thesis AFIT/GLM/LSM/87S-70. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1987 (AD-A186677).

26. Stephenson, Peter. "Telemagic: Telemarketing Package Helps Keep Busy People Organized," InfoWorld, 10: 57 (June 27, 1988).

27. Stock, CMSgt Joeseph P., WRM Plans and Programs HQ SAC. Telephone Interview. Offut AFB NE, 14 November 1989.

28. Study Proposal Project Submission. AFLMC Form 13, Logistics Need ID: 88094 (October 1989).

29. Thomas, Capt Robert S. A Computer Based Data Management System for Air Force War Reserve Materiel (WRM) Vehicle Management. MS Thesis AFIT/GLM/LSM/88S-70. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1988 (AD-A201573).

30. Vesely, Eric G. The Practioner's Blueprint for Logical and Physical Database Design. Englewood Cliffs NJ: Prentice-Hall, 1986.

Vita

Captain Kevin M. Tanzer [redacted] He graduated from Hampton High School in Hampton, New Hampshire in 1981, and attended the University of New Hampshire, graduating with a Bachelor of Art in Chemistry. Upon graduating, he received a reserve commission through the Air Force Reserve Officer Training Corps program. After a 9 month tour of Mather AFB, California, he was the Fuels Management Officer at Holloman AFB, New Mexico within the 833 Supply Squadron. His duties at Holloman AFB focused on support of flying operations and daily contact with maintenance and operations for two wings of tactical aircraft. There he was responsible for directing refueling operations of over 250 aircraft sorties per day, dispensing over 36 million gallons of jet fuel each year, until he entered the School of Systems and Logistics, Air Force Institute of Technology, in May 1989.

[redacted]

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1990	Master's Thesis

4. TITLE AND SUBTITLE
DEVELOPMENT OF SOFTWARE FOR THE BASE-LEVEL WAR RESERVE MATERIELS (WRM) PROGRAM

5. FUNDING NUMBERS

6. AUTHOR(S)
Kevin M. Tanzer, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GLM/LSM/90S-58

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words) This study investigated the requirements for development of a software program for managing War Reserve Materiel (WRM) consumable assests. Areas examined included the requirements for collection, organization, and presentation of data. The research also examined which computer language was appropriate for the successful implementation, and acceptance by the users. A literature review revealed numerous instances of software programs researched and developed to permit management of other WRM program information. The research followed a ten step methodology for developing a database management system application. As part of the methodology, the researcher interviewed eleven experts to determine program requirements. The study developed the prototype software, which was then evaluated by an expert. Following correction of program flaws, a final version of the prototype software was developed. The program was then delivered to fifteen experts for evaluation. The program was accepted by 86.7% of the experts as being a useful tool for managing a WRM program.

14. SUBJECT TERMS
Data Processing, Inventory Control, Computer Programs, Military Equipment, Logistics Planning

15. NUMBER OF PAGES
161

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL